UNIVERSIDADE DA CORUÑA

ESCUELA UNIVERSITARIA POLITÉCNICA

Grado en Ingeniería Electrónica Industrial y Automática

TRABAJO DE FIN DE GRADO

TFG Nº: **770G01A156**

TÍTULO: **IMPLEMENTACIÓN DE UN ALGORITMO DE CONTROL ADAPTATIVO**

AUTOR: **MARÍA REFOJO FERREIRÓS**

TUTOR: **JOSÉ LUIS CASTELEIRO ROCA**
HÉCTOR QUINTIÁN PARDO

FECHA: **JUNIO DE 2019**

Fdo.: EL AUTOR

Fdo.: EL TUTOR

TÍTULO: **IMPLEMENTACIÓN DE UN ALGORITMO DE CONTROL ADAPTATIVO**

ÍNDICE GENERAL

PETICIONARIO: **ESCUELA UNIVERSITARIA POLITÉCNICA**

AVDA. 19 DE FEBREIRO, S/N

15405 - FERROL

FECHA: JUNIO DE 2019

AUTOR: EL ALUMNO

Fdo.: MARÍA REFOJO FERREIRÓS

Índice de figuras

Índice de tablas

TÍTULO: **IMPLEMENTACIÓN DE UN ALGORITMO DE CONTROL ADAPTATIVO**

MEMORIA

PETICIONARIO: **ESCUELA UNIVERSITARIA POLITÉCNICA**

AVDA. 19 DE FEBREIRO, S/N

15405 - FERROL

FECHA: JUNIO DE 2019

AUTOR: EL ALUMNO

Fdo.: MARÍA REFOJO FERREIRÓS

Índice del documento MEMORIA

1 Objeto

En el presente proyecto se abordará la implementación de un algoritmo de control adaptativo basado en el modelo de planta. En la ingeniería de control existen diferentes tipos de algoritmos adaptativos como:

- Reguladores Autosintonizados (STR)
- Control adaptativo con Modelo de Referencia (MRAS)
- Programación de Ganancia

El objeto del presente proyecto es el siguiente:

- Diseño e implementación de un PID mediante ajuste empírico basado en el método Relay-Feedback.
- Diseño y programación de un PID autoajustable basado en el método de mínimos cuadrados recursivos (RLS).
- Creación de una base de datos con los parámetros de la función de transferencia del PID adaptativo para un punto de trabajo conocido.
- Comparación de los resultados obtenidos con ambos reguladores.

Inicialmente se realizará la regulación de un sistema con un PID estándar, usando métodos empíricos como Ziegler-Nichols, Ziegler Nichols con Poca Sobreoscilación, Ziegler-Nichols NO Overshoot y Tyreus Luyben, a través del Método de Relay-Feedback, para diferentes valores de consigna (en modelos de simulación).

A posteriori, una vez obtenida la respuesta con el PID calculado, se procederá a la implementación de un algoritmo de control adaptativo. La finalidad de este algoritmo es el autoajuste en tiempo real de un PID que obtenga las mejores especificaciones de una planta real para cualquier punto de operación.

El primer paso a realizar para el PID adaptativo, consistirá en la implementación del algoritmo RLS, con el cual podrá identificarse la función de transferencia del sistema para posteriormente obtener los parámetros de PID óptimo para el punto de operación sobre el que se esté trabajando.

Una vez obtenidos los PID óptimos, mediante ambos métodos (ajuste empírico y autoajuste), se aplicarán dichos reguladores a los sistemas a controlar, obteniendo las señales más

relevantes del sistema, como son la señal de control u(t), la salida y(t) y el error e(t), con objeto de poder comparar estas señales en ambos reguladores.

La importancia del PID adaptativo es que irá adaptando los parámetros del regulador con cada muestra, es decir, irá adaptando los pesos del PID en función de la salida y(t) y la señal de control u(t) obtenidos para cada muestra.

2 Alcance

A lo largo del proyecto se realizarán los procedimientos descritos a continuación para la implementación del algoritmo de control adaptativo.

- **Estudio del algoritmo RLS para identificación en tiempo real**. Para la identificación de la función de transferencia del sistema se hará uso del algoritmo de mínimos cuadrados recursivo (RLS), por lo que antes de ser implementado se debe estudiar su funcionamiento.

- **Creación de las funciones necesarias para ejecutar el algoritmo de identificación en tiempo real con diferentes configuraciones de la función de transferencia.** Se ha de realizar la identificación en tiempo real óptima de la planta a controlar. Es decir, el orden del sistema, junto con la identificación de los coeficientes del mismo.

  A pesar de que el RLS implementado permitiría la identificación del sistema usando cualquier orden, éste se limitará lo necesario para la sintonización del PID.

- **Realización de pruebas para corroborar el correcto funcionamiento de los PID obtenidos mediante simulación** A pesar de no ser este un punto primordial en el alcance del proyecto, debido a motivos externos (obras en los laboratorios que hicieron no poder disponer de las plantas de nivel) los resultados mostrados están basados en simulación.

  Las pruebas realizadas en simulación se han realizado mediante funciones de transferencia de las plantas de nivel que generan un comportamiento muy similar a las reales

- **Creación de una base de datos de funciones de transferencia identificadas para distintos puntos de operación, cuando se realicen nuevos procedimientos de identificación**. Se ha de crear una base de datos con los valores obtenidos de los parámetros del regulador y señal de control para un punto de consigna conocido, es decir, si se opta por un nivel de agua del 50 % se deben recoger todos los datos calculados para ese punto. De este modo si en una identificación a posteriori se elije el mismo valor de consigna no será necesario realizar todos los cálculos, ya que los datos serán recogidos de la base de datos creada.

- **Implementación de un regulador adaptativo basado en el modelo de planta**. Conocido el proceso de implementación del algoritmo RLS, se debe implementar un regulador PID adaptativo sobre una planta de nivel del "Laboratorio de Optimización y Control", para cualquier punto de consigna 0-100 %.

- **Comparación de los resultados entre el regulador PID estándar y el autoajustable, para el mismo punto de operación.** Se debe implementar un PID estándar para comparar ambos métodos. El ajuste del regulador PID estándar se basará en métodos empíricos (Ziegler-Nichols o Tyreus Luyben) partiendo de los parámetros obtenidos tras la aplicación del sistema Relay-Feedback.

Tanto en el PID adaptativo como en el PID estándar, se graficarán los resultados obtenidos en términos de señal de control, salida del sistema y error del mismo para diversos valores de la señal de consigna .

También se realizarán pruebas con reguladores optimizados para distintos puntos de operación.

3 Antecedentes

La Ingeniería de Control está relacionada con un gran número de campos de la ciencia, desde las matemáticas hasta la biología. Se centra en el control de los procesos dinámicos de manera que las salidas de estos procesos se asemejen lo máximo posible a un comportamiento definido anteriormente, es decir, al comportamiento deseado.

A lo largo de los años, la Ingeniería de Control ha crecido favoreciendo el desarrollo de las actividades en la industria, ocupándose del control de variables como caudal, temperatura, presión, etc.; y con gran importancia en las comunicaciones industriales y control remoto.

En la universidad en la que el proyecto ha sido desarrollado, Escuela Universitaria Politécnica de Ferrol (UDC), "Laboratorio de Optimización y Control"; se cuenta con varias plantas de nivel y hornos para diseño de los diferentes tipos de reguladores. El presente proyecto ha sido realizado en una planta de nivel.

3.1. Planta de Nivel

Es de gran importancia diferenciar las partes de la planta en un diagrama de control como el que se muestra a continuación

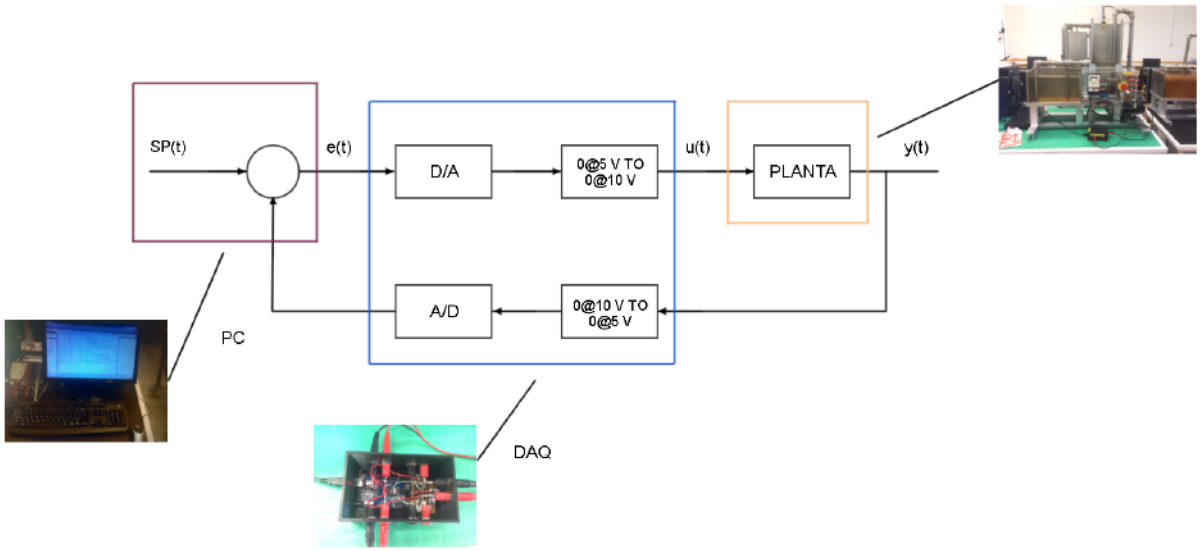


Figura 3.1.0.1 – Diagrama de Control de la Planta de Laboratorio

La planta de nivel sobre la que se ha realizado el proyecto está ubicada en el "Laboratorio de Optimización y Control"de la EUP y se muestra en la figura 3.1.0.2. Con ella se han realizado todas las pruebas necesarias para cumplir todos los puntos nombrados en el alcance.

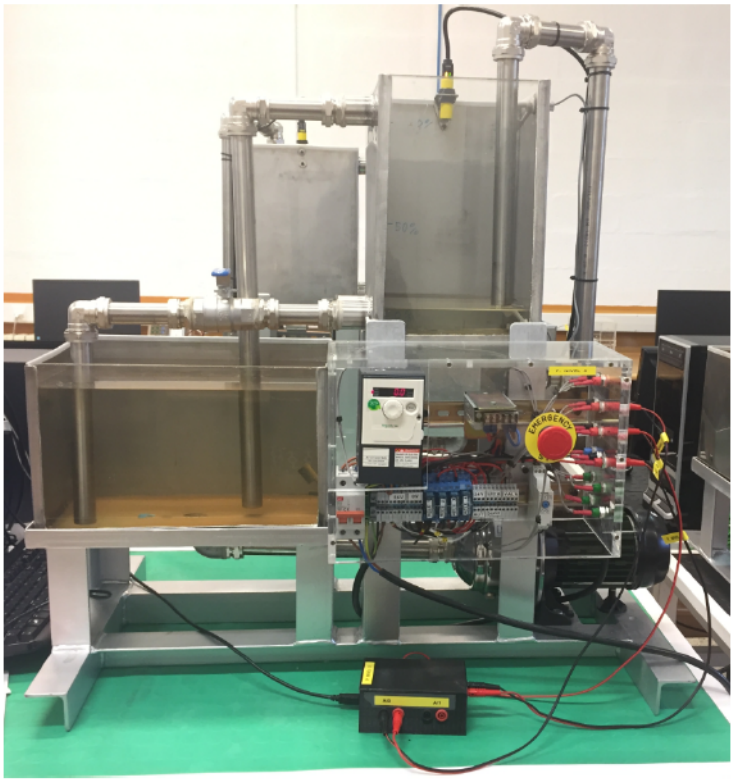


Figura 3.1.0.2 – Planta de Control de Nivel del "Laboratorio de Optimización y Control"(EUP)

La planta consta de dos depósitos de agua, uno en la parte inferior y otro en la parte superior. El depósito de la parte inferior es el que alimenta al superior y éste es el que debe mantener el nivel de agua elegido por el usuario.

Como se puede observar en la figura 3.1.0.2 el depósito inferior tiene mayor capacidad que el superior, protegiendo de este modo la bomba centrífuga y evitando que el depósito se quede sin agua. Para vaciar el depósito superior existen varias opciones:

- Por la bomba centrífuga debido a la propia construcción de las tuberías y la bomba en esta planta, si el tubo del deposito superior desaguara a una altura mayor que la del nivel del agua, éste ya no sería vaciado por la bomba.

- A través de una válvula manual situada en un tubo que va desde el depósito superior al inferior

El funcionamiento de la bomba centrífuga es controlado por un variador de velocidad mediante el software de $MatLab$.

A parte de todo lo mencionado anteriormente, la planta también consta de una seta de emergencia, por si fuese necesario pararla debido a algún problema, como podría ser el desbordamiento del depósito superior, por una programación fallida, o por el mal funcionamiento del sensor que detecta el nivel del agua.

Se han divido en diferentes subsecciones los elementos de los que se compone la planta.

3.1.1. Arduino como DAQ

La adquisición de datos (DAQ) consiste en la medición de fenómenos eléctricos o físicos como voltaje, presión, temperatura, etc. Este sistema consiste de sensores hardware y un PC en el que se almacenan los datos y se realiza la comunicación entre sensores y actuadores. Proporcionan una solución de medidas más potente, flexible y rentable.

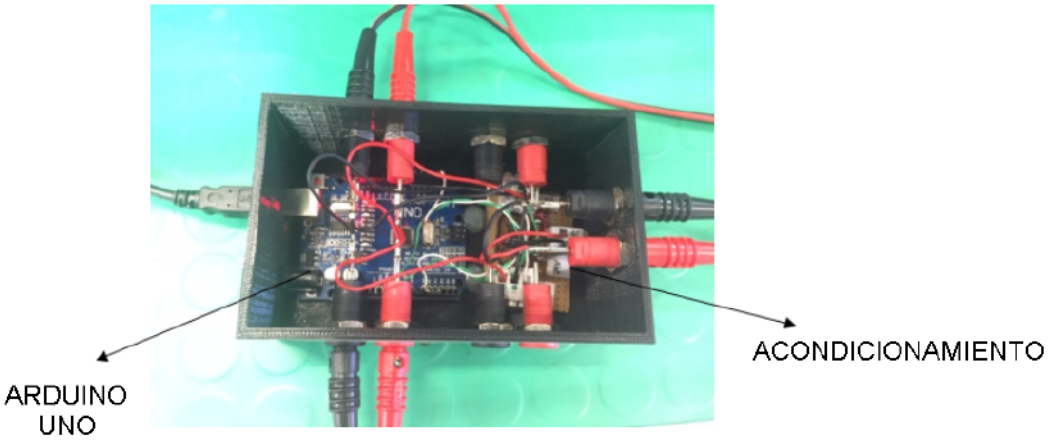


Figura 3.1.1.1 – Arduino como DAQ

Arduino es una plataforma de desarrollo basada en una placa electrónica de hardware libre, que en este caso será usada como sistema de adquisición de datos para la recogida de datos, comprobación y normalización.

Para poder utilizar el arduino como tarjeta de adquisición de datos, es necesario un circuito de acondicionamiento de la señal (ver figura 3.1.1.1), de modo que se conviertan los niveles de tensión generador por el sensor (0 - 10 V) en niveles de tensión admisible en las entradas de arduino (0 - 5 V).

También será necesario acondicionar las salidas del arduino (señal PWM 0 - 5 V) para adaptar los niveles de tensión a los requeridos por el actuador (continua niveles 0 - 10 V).

Por lo tanto el arduino actúa como interfaz de comunicación entre el PC y la planta.

3.1.2. Bomba centrífuga [v]

La bomba centrífuga es la encargada de llevar el agua del depósito inferior al superior.

La bomba de la planta es el modelo MULTINOX-N80/36-T de SACI. Al consultar su catálogo se describe como una electrobomba centrífuga multicelular ideal para formar parte de equipos de presión de mediano y pequeño caudal. Sus principales características son las siguientes:

- Frecuencia de alimentación: 50 Hz
- Potencia máxima de funcionamiento: 0,6 kW
- Motor asíncrono cerrado de ventilación externa con grado de protección IP-44
- Intensidad nominal: 1,2 A
- Caudal máximo: 80 l/min
- Régimen de trabajo: 2850 rpm
- Altura de columna máxima: 35 m
- Tensión de alimentación trifásica 230/400 V
- Máxima temperatura del agua: 50 ºC

Figura 3.1.2.1 – Bomba de la Planta de Control

3.1.3. Variador de Velocidad [vi]

El variador de velocidad de la planta de nivel es el modelo Altivar 312 - ATV312H075M2 de la casa SCHENEIDER y es el encargado de controlar el funcionamiento de la bomba (figura 3.1.3.1). Está destinado para motores asíncronos trifásicos, alimentado a una tensión monofásica de 200-240 V, una potencia máxima de 0,75 kW y frecuencia de alimentación 50-60 Hz. Tiene seis entradas lógicas, tres entradas analógicas, una salida lógica/analógica y dos salidas relé.

Sus principales características son las siguientes:

- Rampas de aceleración y deceleración: Lineales, en forma de S, en forma de U o personalizadas
- Frenado hasta detención
- Protecciones
  - Térmica contra el calentamiento
  - Contra cortocircuitos entre fases del motor
  - De pérdida de fase de entrada, para suministro trifásico
  - Contra roturas de fase del motor
  - Contra sobrecorriente entre salida del motor y tierra

Figura 3.1.3.1 – Variador de Velocidad de la planta de nivel

3.1.4. Sensor Ultrasonidos [viii]

El sensor del que dispone la planta de nivel es de la marca BANNER, modelo U-GAGE S18UUA. Éste es el encargado de detectar el nivel del líquido en tiempo real, en el depósito superior.

Para la medición correcta del sensor se ha determinado un rango, máximo y mínimo en el depósito superior, que irá del 0-100 %. El sensor emite ondas de ultrasonidos que rebotan en el agua y éstas vuelven a ser recibidas por el sensor, midiendo el tiempo transcurrido entre la emisión y recepción. Éste contiene dos LEDs de estado bicolor, uno verde indica que el agua está dentro del rango de detección y el rojo que el objetivo está fuera de este rango.

Tiene como características principales:

- Rango de Sensibilidad: 30 mm - 300 mm
- Tensión de Salida: 0 V - 10 V (DC)
- Tensión de Alimentación: 10 V - 30 V
- Frecuencia de conmutación: 300 kHz
- Rango de Temperaturas de trabajo: -20 ºC a 60 ºC

Figura 3.1.4.1 – Sensor Ultrasonidos de la planta de Control

3.2. Comunicación con la planta de nivel

Para poder ejecutar la programación del presente proyecto, el primer paso será realizar la comunicación con la planta de nivel.

Esta comunicación se realizará como ya se ha comentado a través de una DAQ, en este caso un Arduino, para lo que se usarán una serie de $Scripts$ para que la comunicación sea posible.

Estos $Scripts$ fueron proporcionados por los tutores y por lo tanto no programados para el presente proyecto; su código se puede consultar en 13.1, formado por diferentes funciones:

- DAQ_Start: Inicializa la DAQ.
- DAQ_Write: Genera tensiones en las salidas de la DAQ; indicándose el valor generado en procentaje.
- DAQ_Read: Permite leer los valores de tensión en la entradas de la DAQ, devolviendo a $MatLab$ un valor en porcentaje.
- DAQ_Stop: Finaliza el proceso de comunicación

3.3. Estudio de los Reguladores

En el presente proyecto, basado en un sistema de planta real, se ha utilizado la implementación de los reguladores de control discretos para hacer frente a las posibles perturbaciones o errores que se produzcan en su funcionamiento.

El regulador ha de ser programado en el PC usando como entradas y salidas las señales provenientes de la DAQ explicada en la sección 3.1.1 . El regulador pretende ajustar el funcionamiento real de los sistemas de control que están sometidos a perturbaciones o errores y provocan un funcionamiemto no deseado del sistema. Es decir, intenta que la salida del sistema sea lo más próxima posible a la señal de consigna con la dinámica deseada (set point).

En la actualidad, existen diversos tipos de reguladores, si bien los más empleados en el sector industrial debido a su robustez y sencillez son los tipo P, PI, PID:

- Redes Neuronales
- Fuzzy
- Predictivos
- P, PI, PD y PID

En este caso se ha utilizado un regulador PID para obtener los resultados deseados.

3.3.1. Regulador P, PI, PD, PID

Un regulador de este tipo está formado por diferentes acciones de control, que según sean implementadas pueden dar lugar a otro tipo de reguladores como es el P, PI y PD; que se serán vistos a continuación.

- Acción Proporcional: es la encargada de modificar la ganancia del sistema y modifica el régimen transitorio y permanente. Además disminuye el error, aunque esto puede aumentar la sobreoscilación del sistema, llegando incluso a hacerlo inestable.[4]
- Acción Derivativa: se basa en la variación del error en el tiempo (derivada), por lo que es muy sensible al ruido de la señal de error, pudiendo producir saturaciones en los actuadores cuando el error varía bruscamente. En el plano Z, consiste en añadir un cero al sistema. [4]

  Algún inconveniente importante a destacar es que amplifica las señales de ruido, señales de alta frecuencia, y produce saturaciones en los actuadores cuando la señal de error varía de forma brusca.
- Acción Integral: se basa en la suma del error a lo largo del tiempo (integral), por lo que se elimina el error de posición. En el plano Z consiste en la adición de un polo al sistema en z=1. Su uso aumenta la precisión del sistema con el riesgo de inestabilizarlo, ya que aumenta en uno el tipo del sistema. [4]

3.3.1.1. Regulador Proporcional (P) [4]

El regulador tipo P es el regulador más sencillo a implementar. Aporta una ganancia K_p al sistema, que con su variación consigue realizar una modificación en los régimenes transitorio y permanente de la respuesta de salida.

La salida u(t) de este regulador en tiempo continuo, siendo t el instante actual, será de la siguiente forma:

$$\boxed{u\left(t\right) = K_p \times e\left(t\right)} \tag{3.3.1.1}$$

Siendo e(t) el error, es decir la diferencia entre la salida del sistema y la consigna (SP).

3.3.1.2. Regulador Proporcional-Integral (PI) [4]

En el regulador PI se implementa la acción integral al error e(t). Aporta un polo en z=1 y un cero próximo para minimizar la respuesta dinámica del sistema.

El polo en z=1 hace que aumente el tipo del sistema mejorando así la precisión de éste y corrigiendo el error que presenta en régimen permanente.

Por otro lado el cero ajustable, si se añade, se coloca próximo al polo para evitar que se modifique la respuesta dinámica del sistema.

La salida u(t) de este regulador será de la siguiente forma:

$$\boxed{u\left(t\right) = K_p \left(e\left(t\right) + \frac{1}{T_i}\int_0^t \left(e\left(t\right) \cdot dt\right)\right)} \tag{3.3.1.2}$$

donde T_i es el tiempo integral, que también puede ser expresado como $K_i = 1/T_i$, denominándose esta forma ganancia integral.

3.3.1.3. Regulador Proporcional-Derivativo (PD) [4]

Este tipo de regulador es la implementación de la acción derivativa al error e(k) en tiempo discreto. Aporta un polo en el origen y un cero ajustable al resto del sistema.

El polo en el origen hace que el sistema sea causal, y tenga al menos el mismo número de polos que de ceros, si no el regulador tendría un cero y ningún polo.

Por otro lado, el cero ajustable, proporciona la parte derivativa y determina la respuesta dinámica del sistema.

La salida u(t) de este regulador será de la siguiente forma:

$$\boxed{u\left(t\right) = K_p \left(e\left(t\right) + T_d \cdot \frac{de(t)}{dt}\right)} \tag{3.3.1.3}$$

donde T_d es el tiempo derivativo, que también puede ser expresado como $K_d = T_d$, denominándose esta forma ganancia derivativa.

3.3.1.4. Regulador Proporcional-Integral-Derivativo (PID) [4]

En este último tipo de regulador se tienen en cuenta tanto la acción integral y derivativa sobre la señal de error e(t). Donde la acción derivativa proporciona un cero ajustable y un polo en el origen al sistema y la acción integral un polo en z=1 y un cero próximo. Es el regulador más utilizado junto con el PI, ya que proporciona los mejores resultados.

$$\boxed{u\left(t\right) = K_p \left(e\left(t\right) + \frac{1}{T_i} \int_0^t e\left(t\right) \cdot dt + T_d \cdot \frac{de(t)}{dt} \right)} \tag{3.3.1.4}$$

3.4. Control Adaptativo

El control adaptativo es el método utilizado en el campo del control automático para aquellos sistemas que no pueden ser controlados mediante técnicas clásicas, ya que su forma de operar es cambiante o la planta no está definida de manera adecuada.

Se conoce que los primeros trabajos basados en control adaptativo datan de los años 50 del siglo XX, donde fueron utilizados, permitiendo el desarrollar sistemas automáticos de navegación en aviones y en la industria aeroespacial. Aunque la teoría de este tipo de control, no fue desarrollada hasta los años 70 y 80.

Actualmente se han visto mejorados debido a la aparición de la microelectrónica que hizo de unos reguladores más simples y baratos.

Este control se adapta a los cambios en la dinámica de la planta y a las perturbaciones que se producen en la misma línea, por lo que puede realizar ajustes en tiempo real.

Actualmente existen una gran variedad de algoritmos de control adaptativos, siendo los más utilizados:

- Reguladores Autosintonizados (STR)
- Control Adaptativo con Modelo de Referencia (MRAS)
- Programación de Ganancia

En el presente proyecto el algoritmo de control adaptativo utilizado ha sido el STR explicado en el apartado 7.3

4 Normas y referencias

4.1. Disposiciones legales y normas aplicadas

Las normas aplicadas en el presente proyecto es la normativa de Trabajos de Fin de Grado de la Escuela Universitaria Politécnica.

4.2. Bibliografía

[1] OGATA, KATSUHIKO; *Sistemas de Control en Tiempo Discreto*, 2ª ed., Minnesota, Pearson Educación, (1996).

[2] AGUADO BEHAR,A. Y MARTÍNEZ IRANZO,M.; *Identificación y Control Adaptativo*, 1ª ed., Madrid, Pearson Educación, (2003).

[3] CALVO ROLLE, JOSE L.; *Ajuste empírico de reguladores PID*.

[4] AL-HADITI, BASIL M.; *Análisis y diseño de sistemas de control de nivel*.

4.3. Programas Utilizados

- MATLAB R2014a 32 bits: herramienta de software matemático que ofrece un entorno de desarrollo con un lenguaje de programación propio.
- MiKTex
- TeXstudio

4.4. Otras referencias

[i] *Diseño de Controladores Discretos*, Depto. de Ingeniería de Sistemas y Automática, [Consulta 6 febrero 2019]. Disponible en: http://www.control-class.com/Otros/Apuntes_Ingenieria_Control_Cuarto_IngenieroIndustrial.pdf

[ii] *Sintonización mediante formulación teórica del regulador PID para el control del nivel de agua en depósitos que alimentan aducciones de gran longitud.*. Universidad Politécnica de Valencia. [Consulta 23 enero 2019]. Disponible en: https://riunet.upv.es/bitstream/handle/10251/50781/Trabajo%20fin%20de%20master.pdf?sequence=1

[iii] *Realización de Controladores Discretos*. Sistemas de Control Aplicado. Capítulo 4. Facultad Regional la Rioja. [Consulta 23 enero 2019]. Disponible en: http://www.frlr.utn.edu.ar/archivos/alumnos/electronica/catedras/38-sistemas-de-control-aplicado/Notas_de_C%C3%A1tedra/05_SCA_Cap_4_V7.pdf

[iv] *Identificación Paramétrica discreta*. Esculea Politécnica Nacional Facultad de Ingeniería Elécrica [Consulta 2 marzo 2019]. Disponible en: https://bibdigital.epn.edu.ec/bitstream/15000/11687/1/T199.pdf

[v] *Catálogo General*. Saci pumps [Consulta 2 enero 2019]. Disponible en: http://www.jaenclima.com/catalogos/ficheros/saci.pdf

[vi] *Ficha Técnica del producto ATV12H075M2*. Schneider Eelectric [Consulta 2 enero 2019]. Disponible en: https://www.se.com/cl/es/product/ATV12H075M2/variador-de-velocidad-para-motor-asincrono-con-disipador-0.75kw--1hp--200%E2%80%A6240v-/

[vii] *Datasheet U-GAGE S18U*. BANNER [Consulta 2 enero 2019]. Disponible en: https://www.se.com/cl/es/product/ATV12H075M2/variador-de-velocidad-para-motor-asincrono-con-disipador-0.75kw--1hp--200%E2%80%A6240v-/

[viii] *Datasheet U-GAGE S18U*. BANNER [Consulta 2 enero 2019]. Disponible en: https://www.se.com/cl/es/product/ATV12H075M2/variador-de-velocidad-para-motor-asincrono-con-disipador-0.75kw--1hp--200%E2%80%A6240v-/

[ix] *Apuntes de Ingeniería de Control*. Depto. de Ingeniería de Sistemas y Automática [Consulta 2 enero 2019]. Disponible en: http://www.control-class.com/Otros/Apuntes_Ingenieria_Control_Cuarto_IngenieroIndustrial.pdf

[x] *Aplicación del Control Adaptativo a Procesos Industriales Tipo SISO*. Facultad de Ingeniería Electrónica Bucaramanga [Consulta 2 marzo 2019]. Disponible en: https://repository.upb.edu.co/bitstream/handle/20.500.11912/433/digital_17521.pdf?sequence=1

[xi] *Identificación por Mínimos Cuadrados* Ingeniería de Sistemas y Automática.[Consulta 1 abril 2019]. Disponible en: https://isa.umh.es/asignaturas/mis/ident1.pdf

[xi] *Identificación por Mínimos Cuadrados Recursivos* Ingeniería de Sistemas y Automática.[Consulta 1 abril 2019]. Disponible en: https://isa.umh.es/asignaturas/mis/ident2.pdf

5 Definiciones y abreviaturas

- PID: Proporcional Integral Derivativo
- RLS: Recursive Least Squares (Mínimos Cuadrados Recursivos)
- SP: Set Point
- FT: Función de Transferencia
- DAQ: Data Acquisition Serial
- STR: Self-Tuning Regulator
- MRAS: Model Reference Adaptive Systems
- Regulador P: Regulador Proporcional
- Regulador PI: Regulador Proporcional-Integral
- Regulador PD: Regulador Proporcional-Derivativo
- Regulador PID: Regulador Proporcional-Integral-Derivativo

6 Requisitos de diseño

Se deberán cumplir una serie de requisitos para el diseño del sistema, nombrados a continuación:

- Se debe operar la planta mediante $MatLab$.
- Para la obtención de los resultados es necesaria la realización de pruebas experimentales sobre la planta de nivel o mediante simulación (en el presente proyecto han sido tomadas mediante simulación por razones ya comentadas).
- Para ambos regulador (PID estándar y PID adaptativo), se debe trabajar con diferentes puntos de operación. Es decir, el punto de trabajo puede ir cambiando cada cierto número de muestras en la misma prueba.
- Se deberán realizar las simulaciones de ambos reguladores con los mismos puntos de trabajo para hacer una comparación entre ambos.
- El período de muestreo T es constante en ambos reguladores.
- En la planta real si se aplicasen perturbaciones se realizarían con la apertura de la válvula manual. Por lo contrario, si no existen perturbaciones, ésta válvula manual sólo sería usada para el desagüe del depósito superior.
- En el entorno de simulación, se ha trabajado con funciones de transferencia de grado 1 y 2 del numerador y denominador respectivamente para obtener la máxima similitud con la planta real, ya que para ésta se estaría ante un sistema de segundo orden.
- La obtención de los parámetros del PID estándar se han realizado por ajuste empírico a través del método de Relay-Feedback.
- Se debe garantizar un números de muestras capaz de estabilizar el sistema para los diferentes puntos de consigna en una misma prueba.
- El código de programación debe ser realizado de manera óptima, de modo que si se quisiese utilizar en futuros proyectos pudiese ser reutilizable.

7 Análisis de las soluciones

Actualmente existen diferentes formas de controlar un proceso (temperatura, caudal, presión, etc.), mediante la Ingeniería de Control. Aplicando el marco teórico el control puede realizarse de manera lineal, no lineal, óptimo y robusto; mientras que si se aplica el tipo de control puede ser en lazo abierto, lazo cerrado, regulación o seguimiento de trayectorias.

En el presente proyecto, se deberá realizar el control de una planta de nivel mediante un PC a través de una DAQ (Arduino) que realice de interfaz entre la planta y el PC.

Éste control se ha realizado mediante el método de regulación, el cual requiere de la implementación de uno de los diferentes reguladores de control, ya mencionados en el apartado 3.3, para la obtención de una solución que se acerque lo máximo posible a la deseada.

7.1. Discretización del regulador PID

El regulador PID es el controlador por excelencia, ya que se caracteriza por su fácil aplicación en sistemas en los que no se conoce el modelo matemático.

Éste es una implementación simple basada en la realimentación, que permite reducir los efectos de las perturbaciones haciendo que el sistema se vuelva insensible a las variaciones, como ya se ha comentado anteriormente.

Basándose en la teoría de control de los reguladores PID explicada en detalle en el anexo 10.2, se procedió a la discretización de dicho regulador PID mediante la discretización de $Tustin$, ya que es una de las discretizaciones de PID más empleadas en la actualidad.

Para explicar la descritazión de $Tustin$ se debe partir de la ecuación del regulador en tiempo continuo, para la que haciendo la transforma de $Laplace$ de la función de transferencia del regulador (la salida de éste con respecto a su entrada) se obtiene la ecuación 7.1.0.1 [iii]

$$\boxed{\frac{U(s)}{E(s)} = K_p(1 + \frac{1}{T_i s} + T_d s)} \tag{7.1.0.1}$$

A partir de la ecuación 7.1.0.1, se realiza la discretización por $Tustin$, la cual consiste en sustituir la s de la función de transferencia en $Laplace$ (ecuación 7.1.0.1) por $\frac{2}{T} \cdot \frac{z-1}{z+1}$.

Al sustituir la s en la parte derivativa, se introduce un polo inestable en $z = -1$, de modo que la discretización de la parte derivativa se modificaría a la ecuación 7.1.0.2, que genera un polo estable en el origen.

$$s = \frac{2}{T} \cdot \frac{z-1}{z} \tag{7.1.0.2}$$

De forma que para la ecuación 7.1.0.1, aplicando la discretización da lugar la ecuación 7.1.0.3

$$\boxed{\frac{U(z)}{E(z)} = K_p(1 + \frac{T}{2T_i} \cdot \frac{z+1}{z-1} + \frac{2}{T_d}T \cdot \frac{z-1}{z}) = K_p(1 + \frac{T}{2T_i} \cdot \frac{1+z^{-1}}{1-z^{-1}} + \frac{2}{T_d}T \cdot \frac{1-z^{-1}}{1})} \quad (7.1.0.3)$$

7.1.1. Obtención de la señal de control en ecuación en diferencias [i]

Una vez realizada la discretización del regulador y partiendo de la función de transferencia discretizada del sistema, ecuación 7.1.0.3, operando se llega a una fórmula reducida (ecuación en diferencias equivalente) del siguiente modo, ecuación 7.1.1.1.

Este método para la obtención de la señal de control sería una especie de 'truco' utilizado para implementar de una forma simple, una función de transferencia, además de la base para realizar la identificación por mínimos cuadrados.

Siendo la ecuación 7.1.1.1 la señal de control del regulador a calcular y las ecuaciones 7.1.1.2, 7.1.1.3 y 7.1.1.4 los parámetros de la anterior.

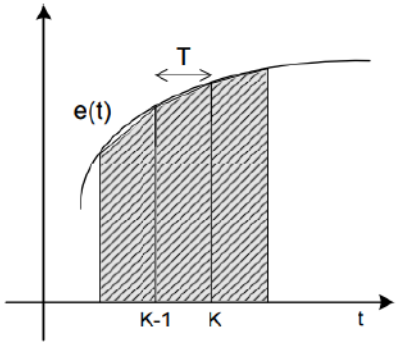


Figura 7.1.1.1 – Aproximaxión del error por el método de Tustin

$$\boxed{u_k - u_{k-1} = q_0 e_k + q_1 e_{k-1} q_2 e_{k-2}} \quad (7.1.1.1)$$

- $$q_0 = K_p(1 + \frac{T}{2T_i} + \frac{T_d}{T}) \quad (7.1.1.2)$$
- $$q_1 = K_p(\frac{T}{2T_i} - 1 - 2\frac{T_d}{T}) \quad (7.1.1.3)$$
- $$q_2 = K_p\frac{T_d}{T} \quad (7.1.1.4)$$

Para la implementación del regulador PID estándar es necesario obtener los parámetros (Kp, Td, Ti) del mismo. El método del Relay-Feedback, sería una opción para la obtención de éstos para un punto de trabajo concreto, explicado en la sección 7.2.1.

Sabiendo que las constantes K_p, K_i y K_d, son las ganancias proporcional y derivativa resperctivamente, necesarias para el cálculo de la señal de control del regulador. Éstas se rigen por las siguientes ecuaciones (donde T es el periodo de muestreo):

[2]

$$\boxed{K_p = -(p_1 + 2p_2)} \quad (7.1.1.5)$$

$$\boxed{K_i = \frac{(p_0 + p_1 + p_1)}{T}} \quad (7.1.1.6)$$

$$\boxed{K_d = p_2 T} \tag{7.1.1.7}$$

7.2. Ajuste empírico de reguladores

Uno de los métodos de ajuste de reguladores más empleados en la actualidad, consiste en la determinación empírica de los parámetros del regulador basado en mediciones sobre la respuesta del sistema en determinadas condiciones de funcionamiento.

En el caso de los sistemas en cadena abierta, dichas, dichas mediciones se realizan sobre la curva de reacción del sistema ante entrada escalón.

En cadena cerrada, puede medirse llevando al sistema a oscilación o bien en caso de no ser posible, puede aplicarse el método del Relay-Feedback.

En ambos casos, una vez realizadas las medidas necesarias sobre la respuesta del sistema, se calcularán los parámetros del regulador en base a ecuaciones empíricas de diversos autores (Ziegler-Nichols y Tyreus Luyben).

7.2.1. Método Relay-Feedback [3]

Método desarrollado por Åström y Hagglünd, que sonsiste en llevar al sistema al estado de oscilación con la incorporación de un relé (figura 7.2.1.1). De este modo se podrá localizar la ganancia crítica (K_c) y el periodo de oscilación sostenida (T_c) del sistema. Parámetros necesarios para realizar los cálculos para obtener los parámetros del regulador (K_p, T_d, T_i) [3].

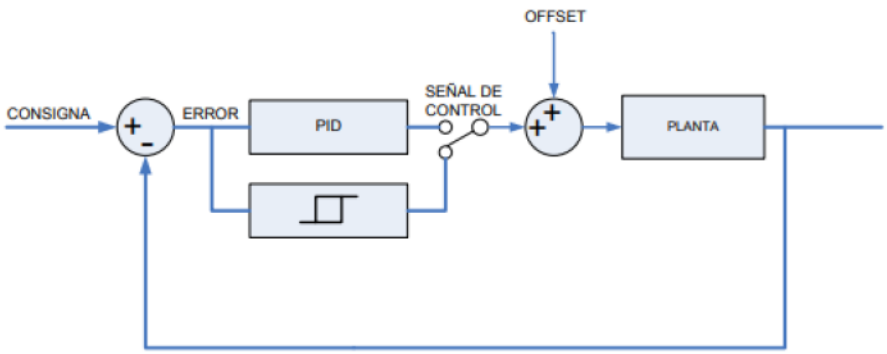


Figura 7.2.1.1 – Diagrama de Bloques Relay-Feedback [3]

Se debe emplear un relé con histérisis con las siguientes características (figura 7.2.1.2)

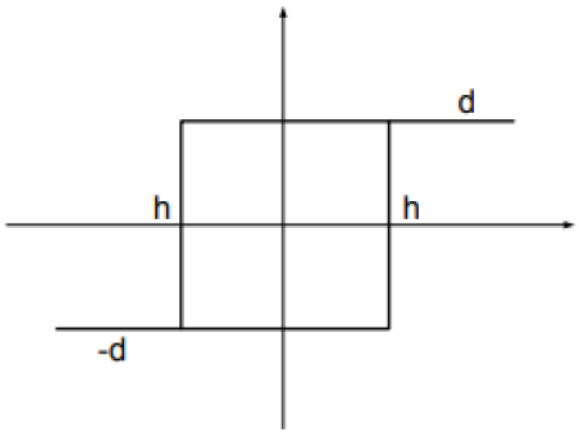


Figura 7.2.1.2 – Histérisis del relé para Relay-Feedback [3]

donde d es la amplitud y h el ancho de la ventana de histéresis.

Una vez realizado el Relay-Feedback se obtendrá una señal oscilante periódica, de la cual se obtendrán los parámetros Tc y a (figura 7.2.1.3).

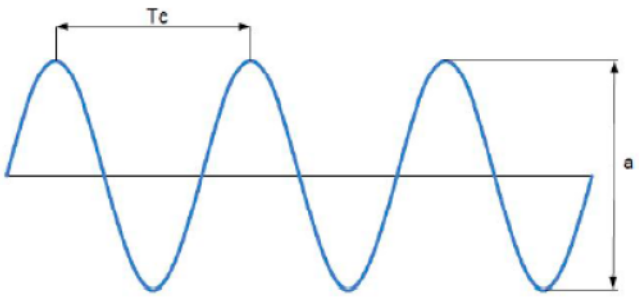


Figura 7.2.1.3 – Histérisis del relé para Relay-Feedback [3]

Como se puede observar en la figura 7.2.1.3, Tc y a se obtienen de la gráfica resultante mientras que para obtener la ganancia crítica (K_c) se debe emplear la fórmula de la ecuación 7.2.1.1 [3].

$$Kc = \frac{4d}{\pi\sqrt{a^2 - h^2}} \tag{7.2.1.1}$$

7.2.2. Obtención de los parámetros del regulador PID (Kp, Td, Ti)

Como ya se ha comentado anteriormente para la obtención de los parámetros del regulador se necesita de la ganancia crítica (K_c) y del periodo de oscilación sostenida (T_c) del sistema. Una vez obtenidos estos dos parámetros se puede proceder al cálculo de los parámetros del PID mediante una serie de fórmulas matemáticas. En el presente proyecto se han utilizado las fórmulas de *Ziegler-Nichols* en cadena cerrada y las de *Tyreus-Luyben*.

7.2.2.1. Ziegler-Nichols [3]

La primera aproximación de las expresiones de *Ziegler-Nichols* en cadena cerrada se muestra en la tabla 7.2.2.1

Parámetro	Expresión
Kp	0.6·Kc
Ti	0.5·Tc
Td	0.125·Tc

Tabla 7.2.2.1 – Fórmulas *Ziegler-Nichols* en cadena cerrada [3]

Esto es aplicable cuando se cumple $2<k\cdot K_c<20$ siendo k la ganancia del proceso, que es la cantidad de variación en la salida provocada por una variación en la entrada y define la sensibilidad de la planta, y K_c la ganancia crítica calculada mediante el método de Relay-Feedback [3].

El método *Ziegler-Nichols* es mejorable en ciertos aspectos, por lo que a raíz de esta base ciertos autores han propuesto nuevas expresiones para mejorar ciertas especificaciones del sistema [3].

1. ***Ziegler-Nichols* con poca sobreoscilación** [3]

   En esta modificación se intenta reducir la sobreoscilación del sistema, como su propio nombre indica, y sus nuevas expresiones se pueden ver en la tabla 7.2.2.2

Parámetro	Expresión
Kp	0.33·Kc
Ti	Tc/2
Td	Tc/3

Tabla 7.2.2.2 – Fórmulas *Ziegler-Nichols* en cadena cerrada con poca sobreoscilación [3]

2. ***Ziegler-Nichols* sin sobreoscilación** [3]

   En esta nueva modificación, tabla 7.2.2.3, hay una mejora de la sobreoscilación con respecto a la primera aproximación y a la modificación anterior.

Parámetro	Expresión
Kp	0.2·Kc
Ti	Tc
Td	Tc/3

Tabla 7.2.2.3 – Fórmulas *Ziegler-Nichols* en cadena cerrada sin sobreoscilación [3]

7.2.2.2. Tyreus-Luyben [3]

Para este método el rango de aplicación es el mismo que el mostrado para las expresiones de *Ziegler-Nichols* de la tabla 7.2.2.1. Mostrándose sus expresiones en la tabla 7.2.2.4 .

Parámetro	Expresión
Kp	0.45·Kc
Ti	2.2·Tc
Td	Tc/6.3

Tabla 7.2.2.4 – Fórmulas *Tyreus-Luyben* [3]

7.3. Reguladores Autosintonizados (STR) [ix] [x]

El regulador STR es un algoritmo de control adaptativo en el que se recogen las variables de entrada y salida del sistema, para realizar una estimación de los parámetros del mismo de forma recursiva. Los cambios transcurridos en el tiempo en el sistema real son linealizados mediante un proceso en el que los parámetros son actualizados de forma recursiva, ajustándose así lo máximo posibe al sistema real.

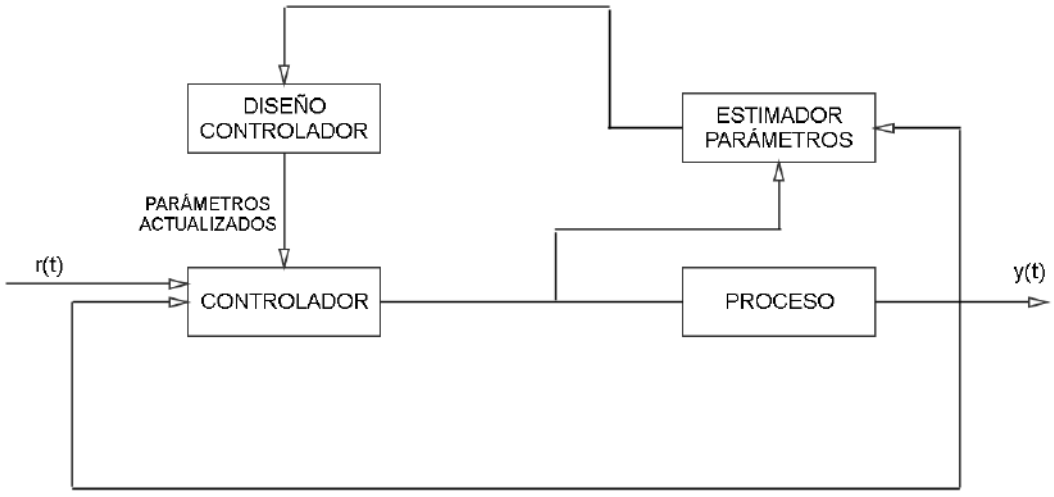


Figura 7.3.0.1 – Diagrama de Bloques de Regulador Autosintonizado (STR)

Como se puede observar en la figura 7.3.0.1, este tipo de regulador se puede dividir en dos lazos. Un primer lazo, el proceso con realimentación y un segundo lazo, en el que se estiman los parámetros del sistema y el bloque donde se realizan los cálculos del regulador.

El bloque *DISEÑO CONTROLADOR* en la figura 7.3.0.1, resuelve en tiempo real el problema del diseño del sistema con parámetros conocidos. Este tipo de regulador consta de un estimador recursivo que se puede implementar por diversas técnicas como:

- Método de Mínimos Cuadrados
- Método de Mínimos Cuadrados Recursivos (RLS)
- Mínimos Cuadrados Extendidos
- Método Maximun Likelihood (Método de Máxima Verosimilitud)

En este caso se ha utilizado el Método de Mínimos Cuadrados Recursivos (RLS) para realizar la estimación.

7.3.1. Algoritmo de Mínimo Cuadrados [iv] [xi]

Para diseñar el regulador a implementar en el sistema de control, se debe disponer de su función de transferencia. El método más sencillo para realizar la estimación de los parámetros de esa función de transferencia es el algoritmo de Mínimos Cuadrados.

El algoritmo Mínimos Cuadrados consiste en la obtención de los parámetros de un sistema, donde la función objetivo será minimizar la diferencia entre la salida real del sistema y(k) y la salida estimada $\hat{y}$(k). Esto es realizable gracias a la ecuación en diferencias para el cálculo de la salida de un sistema definido con una determinada función de transferencia discreta, ya que se puede convertir la identificación de los parámetros de dicha función en una válida para el uso de mínimos cuadrados.

Es una técnica simple si el modelo tiene la propiedad de ser lineal en sus parámetros, así la estimación de éstos es calculada de forma analítica. Este método es un estimador experimental para la construcción de modelos que puede ser estructurado en los siguientes pasos:

- **Planificación experimental**: el sistema a identificar es excitado por una señal de entrada y se registra para cada instante la entrada y salida producida.
- **Selección de la estructura del modelo**: atendiendo a la salida de la planta obtenida en el apartado anterior, se puede deducir la función de transferencia a identificar (primer orden, segundo orden, retardos, etc.).
- **Estimación de parámetros**: una vez conocida la estructura del sistema, se obtienen el valor de los parámetros desconocidos.
- **Validación**: comprobar si el sistema responde de la manera esperada.

En relación a los pasos antes nombrados, el primer paso es considerar un modelo genérico, ecuación 7.3.1.1 donde los parámetros a estimar serán $a_1, a_2, ..., a_n, b_0, b_1, ..., b_n$.

$$\boxed{G(z) = \frac{b_0 + b_1 z^{-1} + ... + b_m z^{-m}}{1 + a_1 z^{-1} + ... + a_m z^{-m}}} \tag{7.3.1.1}$$

Para obtener esos parámetros se debe realizar una estimación de la salida de la forma (ecuación 7.3.1.2).

$$\boxed{\hat{y}(k+1) = {x_{k+1}}^T \cdot \theta} \tag{7.3.1.2}$$

donde θ, es el vector de parámetros a estimar y x_{k+1} el vector (ecuación 7.3.1.3), donde y es la salida del sistema, u la señal de control (entrada a la planta), d el retardo que presenta el sistema y k el instante actual.

$$\boxed{x_{k+1} = [-y(k), -y(k-1), ..., -y(k-n+1), u(k-d), ..., u(k-m+1-d)]} \tag{7.3.1.3}$$

El algoritmo Mínimos Cuadrados es aquel cuyos valores de los parámetros minimicen la suma de los cuadrados de los errores, siendo el error e_{k+1} la diferencia entre la salida real y la estimada (ecuación 7.3.1.4), y los parámetros se obtienen del siguiente modo, ecuación 7.3.1.5

$$\boxed{e_{k+1} = y_{k+1} - \hat{y}_{k+1}} \tag{7.3.1.4}$$

$$\boxed{\hat{\theta} = (X^T X)^{-1} X^T Y} \tag{7.3.1.5}$$

Siendo:

- θ: el vector de parámetros estimados para el cálculo de la función de transferencia.
- X: matriz de entradas y salidas del sistema real determinada por la ecuación 7.3.1.3. Ésta matriz aumenta su tamaño en función del número de iteraciones que sean llevadas a cabo, es decir, para los datos de una iteración se tendría una fila; por lo que para los datos de 1000 iteraciones se tendría una matriz de 1000 filas, dando lugar a una matriz 1000x1000.
- Y: el vector formado por las salidas estimadas del sistema, calculadas en cada muestra con la ecuación 7.3.1.2.

Este tipo de algoritmo presenta un problema a tener en cuenta para la realización de este trabajo, ya que como pudimos observar hace uso de la inversa de una matriz, $(X^T X)^{-1}$, para su resolución. Si ésta es excesivamente grande, su cálculo se vuelve complejo; por lo que el algoritmo Mínimos Cuadrados no es válido para estimación de parámetros en tiempo real, ya que cada muestra haría crecer la matriz. Además, podría darse el caso de la no existencia de dicha matriz inversa, dando lugar a un problema de convergencia de los pesos a calcular.

La solución a este problema es la utilización del Algoritmo de Mínimos Cuadrados Recursivos (RLS), explicado a continuación, ya que no utiliza la inversa de una matriz.

7.3.2. Algoritmo de Mínimo Cuadrados Recursivos (RLS) [iv] [xi]

El algoritmo de mínimos cuadrados recursivos es uno de los métodos más utilizados para la identificación de parámetros, ya que como se ha dicho en el apartado anterior no utiliza la inversa de una matriz para sus cálculos por lo que hace posible la estimación en tiempo real. Siendo así una variante del algoritmo de Mínimos Cuadrados y sujeto a sus misma restricciones nombradas en el apartado 7.3.1

El algoritmo recursivo utiliza los datos del paso anterior para el actual, de este modo el cálculo de los parámetros es como se indica en la ecuación 7.3.2.1.

$$\boxed{\hat{\theta}_{k+1} = \hat{\theta}_k + correcion} \tag{7.3.2.1}$$

Siendo:

- θ_{k+1} el vector de parámetros estimados en un instante
- θ_k el vector de parámetros estimados en el instante anterior
- La correción se corresponde con el ajuste de la nueva estimación basada en los valores obtenidos en el instante actual.

Para su implementación se debe partir de una función de transferencia genérica, con grados en el numerador y denominador cualesquiera, siendo esta función de transferencia la mostrada en la ecuación 7.3.2.2.

$$\boxed{G(z) = \frac{b_0 + b_1 z^{-1} + ... + b_m z^{-m}}{1 + a_1 z^{-1} + ... + a_m z^{-m}}} \tag{7.3.2.2}$$

De esta función de transferencia genérica, ecuación 7.3.2.2, en función del grado con el que se vaya a trabajar se pueden obtener las diferentes funciones de transferencia, ecuación 7.3.2.3 y 7.3.2.4, siendo éstas de segundo grado en el denominador, entre otras.

- Función de Transferencia de Grado 0

$$\boxed{G(z) = \frac{b0}{z^2 + a_1 z + a_0}} \tag{7.3.2.3}$$

- Función de Transferencia de Grado 1

$$\boxed{G(z) = \frac{b_0 + b_1 z}{z^2 + a_1 z + a_0}} \tag{7.3.2.4}$$

Elegidos los grados del numerador y denominador se debe desarrollar la función de transferencia en una ecuación en diferencias. Para ejemplificarlo, se supondrá que los grados del numerador y denominador son de grado 1 y 2 respectivamente (función de transferencia elegida en el presente proyecto).

Se tiene entonces una función de transferencia del siguiente modo:

$$\boxed{G(z) = \frac{Y(z)}{U(z)} = \frac{b_0 + b_1 z}{z^2 + a_1 z + a_0} = \frac{b_0 + b_1 z^{-1}}{1 + a_1 z^{-1} + a_0 z^{-2}}} \quad (7.3.2.5)$$

$$\boxed{y_k + a_1 y_{k-1} + a_0 y_{k-2} = b_1 u_{k-1} + b_0 u_{k-2}} \quad (7.3.2.6)$$

$$\boxed{y_k = b_1 u_{k-1} + b_0 u_{k-2} - a_1 y_{k-1} - a_0 y_{k-2}} \quad (7.3.2.7)$$

$$\boxed{y = x^T \cdot \theta} \quad (7.3.2.8)$$

Desarrollando la ecuación en diferencias, ecuación 7.3.2.6, y despejando la salida y_k se obtiene la ecuación 7.3.2.7. Esta ecuación 7.3.2.7 se puede escribir de forma matricial, ecuación 7.3.2.8, donde:

- y: valor actual de la salida y_k.
- x: vector de estados anteriores de entradas y salidas.

$$\boxed{x = [u_{k-1}; -y_{k-1}; -y_{k-2}]} \quad (7.3.2.9)$$

- θ: vector con los parámetros $b1$, $a1$ y $a2$ de la función de transferencia G(z).

Además de los parámetros nombrados anteriormente, este algoritmo necesita una matriz P para el cálculo de los parámetros de θ que debe ser inicializada con valores aleatorios y debe tener la misma dimensión que el vector θ.

La solución del algoritmo sigue una serie de pasos:

1. Definir los valores de P_k y $\hat{\theta_k}$. La matriz P se inicializa con un valor aleatorio y θ si no se conoce nada a cerca de los parámetros será inicializada del mismo modo que P_k, con un valor aleatorio .

   Es necesario inicializar la matriz P_k y el vector de parámetros θ_k, ya que en el instante inicial necesitamos un valor de éstos para el cálculo de la matriz de ganancias $K_{(k+1)}$. Una vez pasada la primera muestras estos parámetros se irán calculando según las ecuaciones 7.3.2.13 y 7.3.2.14 en función de los datos del instante anterior.

2. Obtención de las medidas de las señales de entrada (u_{k+1}) y salida (y_{k+1})

3. Formar el vector regresor x_{k+1} que tiene la siguiente forma

$$\boxed{x = [u_{k-1}; -y_{k-1}; -y_{k-2}]} \quad (7.3.2.10)$$

4. Matriz de ganancias $K_{(k+1)}$

$$K_{(k+1)} = \frac{P_k}{1 + (x_{k+1})^T \cdot P_{k+1} \cdot x_{k+1}} \tag{7.3.2.11}$$

5. Cálculo del error de predicción $e_{(k+1)}$

$$e_{(k+1)} = y_{k+1} - (x_{k+1})^T \cdot \hat{\theta}_k \tag{7.3.2.12}$$

6. Cálculo de los parámeros estimados θ_{k+1}

$$\hat{\theta}_{(k+1)} = \hat{\theta}_k + K_{(k+1)} \cdot x_{k+1} \cdot e_{(k+1)} \tag{7.3.2.13}$$

7. Cálculo de la matriz P para el siguiente instante

$$P_{(k+1)} = P_k - \frac{P_k \cdot x_{k+1} \cdot (x_{k+1})^T \cdot P_k}{1 + (x_{k+1})^T \cdot P_k \cdot x_{k+1}} \tag{7.3.2.14}$$

8. Vuelta al punto 2 para el siguiente muestreo

El RLS presenta un problema que debe ser corregido, siendo este la incapacidad de detectar cambios en los parámetros a lo largo del tiempo. Esto se debe a que la matriz P se hace muy pequeña.

La solución a este problema es la implementación de un factor de olvido (λ) al algoritmo recursivo. De esta forma se pondera más a las muestras más recientes permitiendo detectar un cambio en los parámetros.

Las fórmulas del RLS con factor de olvido son similares a las mostradas anteriormente excepto las ecuaciones 7.3.2.14 y 7.3.2.11, que son las que añaden (λ), el factor de olvido, y el procedimiento de aplicación es el mismo que el anterior.

Ahora las nuevas fórmulas para $P_{(k+1)}$ y $K_{(k+1)}$ son:

$$K_{(k+1)} = \frac{P_k}{\lambda + (x_{k+1})^T * P_{k+1} * x_{k+1}} \tag{7.3.2.15}$$

$$P_{(k+1)} = \frac{1}{\lambda}(P_k - \frac{P_k * x_{k+1} * (x_{k+1})^T * P_k}{\lambda + (x_{k+1})^T * P_k * x_{k+1}}) \tag{7.3.2.16}$$

Es importante tener en cuenta el valor del factor de olvido que debe ser $0<\lambda<1$, ya que dependiendo de su valor se obtienen diferentes comportamientos.

- λ pequeño: El sistema es más cambiante y puede introducir ruido en el sistema.
- λ grande: Existen problemas a la hora de detectar cambios en los parámetros, ya que la identificación es más lenta ante el cambio.

Normalmente el valor del factor de olvido que se suele elegir rondará entre $0.9<\lambda<0.98$

8 Resultados finales

Todos los resultados mostrados a continuación fueron realizados mediante simulación, ya que no se han podido realizar las pruebas pertinentes en la planata real, como ya se ha explicado anteriormente.

8.1. Regulador PID estándar

Se ha realizado un PID estándar en la planta de nivel para realizar una comparación de los resultados entre este PID y un PID adaptativo usando el algoritmo RLS.

8.1.1. Cálculo de los parámetros Tc y Kc

El primer paso para la realización de este PID ha sido la obtención de los parámetros del regulador, que se ha realizado mediante el método del Relay-Feedback explicado en la sección 7.2.1.

Se han realizado varias gráficas con el método del Relay-Feedback, ya que dependiendo del punto de trabajo seleccionado se obtendrán diferentes valores para el cálculo de parámetros del regulador.

Se puede ver la gráfica generada por la función Relay-Feedback, de la que se puede ver el código en 13.3.2 para un punto de trabajo del 40 %, en la figura 8.1.1.1

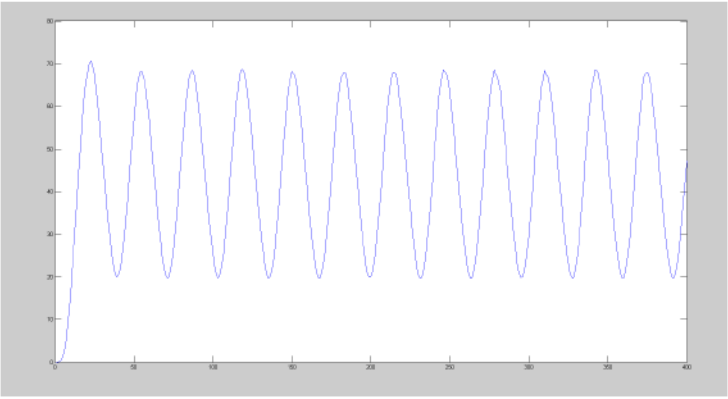


Figura 8.1.1.1 – Gráfica Relay-Feedback para punto de trabajo 40 %

Como se mencionaba en la sección 7.2.1, T_c y a se obtienen directamente a partir de la figura 8.1.1.1 y K_c mediante la fórmula 7.2.1.1 que sirvirán para el cálculo de los parámetros del PID.

Siendo para este punto de trabajo (tabla 8.1.1.1):

Tc=31.67
a=50.62
Kc=1.265

Tabla 8.1.1.1 – Parámetros 40 % *Relay-Feedback*

Cabe decir que para la obtención de T_c y a a partir de la figura 8.1.1.1 se han recogido diferentes valores de estos parámetros para obtener así un valor medio y consiguir unos valores más próximos a lo representado. Esto se ha realizado para cada punto de trabajo con sus respectivas gráficas de Relay-Feedback.

Los resultados obtenidos se pueden consultar en el anexo 11.1.1.1 y el código de su función en el anexo 13.3.2.

8.1.2. Cálculo de los parámetros del regulador Kp, Ti y Td mediante ajuste empírico

Una vez obtenidos los parámetros K_c y T_c, se procede al cálculo de los parámetros del regulador pudiendo elegir diferentes formas para su obtención:

- Ziegler-Nichols
  - Primera aproximación de Ziegler-Nichols
  - Ziegler-Nichols con poca sobreoscilación
  - Ziegler-Nichols sin sobreoscilación
- Tyreus-Luyben

Los resultados obtenidos de los parámetros del regulador para un 40 % de punto de trabajo son los mostrados en la tabla 8.1.2.1 y el código de su función en 13.3.3.

Parámetros	ZN	ZN Poca Sobre	ZN No Over	TL
Kp	0.76	0.42	0.25	0.57
Ti	15.79	15.79	31.57	69.45
Td	3.95	10.52	10.52	5.01

Tabla 8.1.2.1 – Parámetros Regulador 40 % punto de trabajo

Una vez se han calculado los parámetros del regulador necesarios para la señal de control, se cogerá el valor de la consigna de un vector con diferentes valores de consigna.

A partir de este punto el cálculo de la señal de control seguirá los siguientes pasos:

1. Se escribe la señal de control $u(t)$ mediante DAQ_Write.
2. Se lee la variable de proceso de la planta mediante DAQ_Read.

3. Se modifica el SP cada cierto número de muestras. Ésto se debe realizar antes del cálculo del error para que en el momento que cambie el SP, el próximo cálculo del error ya se realiza en base a la nueva consigna.

4. Se recalculan los valores del error $e(t)$ antes de calcular la señal de control $u(t)$.

5. Se calcula la señal $u(t)$.

Para el cálculo de la señal de control se ha creado una función que se puede consultar en el anexo 13.3.4 y todo lo anterior se ha implementado a un programa principal (anexo 13.3.1).

8.1.3. Cálculo de la señal de control

Por último se ha de calcular la señal de control del regulador PID estándar. Cálculo que se ha realizado con la ecuación discretizada del regulador PID, del mismo modo que para el PID adaptativo.

El código de esta función se puede consultar en la sección 13.3.4 y los pasos a seguir para su cálculo son:

1. Se deben pasar los valores de K_p, T_i y T_d obtenidos a partir del método elegido (Ziegler-Nichols, Ziegler-Nichols con Poca Sobreoscilación, Ziegler-Nichols No Overshoot y Tyreus Luyben), así como la señal de control y el error en ese instante.

2. Basándose en la ecuación 7.1.1.1, se deben calcular los parámetros q_o, q_1 y q_2. Parámetros que vienen determinados por las ecuaciones 7.1.1.2, 7.1.1.3 y 7.1.1.4.

3. Calculados los parámetros, se aplica la ecuación 7.1.1.1 y se obtiene la nueval señal de control del regulador.

   Se debe limitar la señal de control en ambos reguladores (PID estándar y PID adaptativo), ya que se trabaja con un rango de nivel de consigna del 0 - 100 $\%$ (eliminando así el efecto de acumular mucha parte integral, anti-windup). Su cálculo puede dar como resultado un valor por encima o por debajo de este rango, por lo que se debe limitar la señal. Esta limitación se puede consultar en el anexo 13.4.4 y 13.3.4.

8.1.4. Resultados Obtenidos con el PID estándar

A continuación se muestran los resultados obtenidos para el PID estándar con el 40 % de punto de trabajo y diferentes valores de consigna.

En función de la aproximación elegida (Ziegler-Nichols, Ziegler-Nichols poca sobreoscilación, Ziegler-Nichols No Overshoot y $TyreusLuyben$) se obtuvieron los diferentes resultados, correspondiendo cada trazado a lo siguiente:

- Plot superior: Valor real de la variable del proceso (la salida del sistema, línea azul) y el punto de consigna que debe alcanzar ésta salida (línea a trazos negra).
- Plot intermedio: Se muestra el error que se fue obteniendo durante el proceso.
- Plot inferior: Se muestra la señal de control

Siendo las simulaciones:

- Ziegler-Nichols

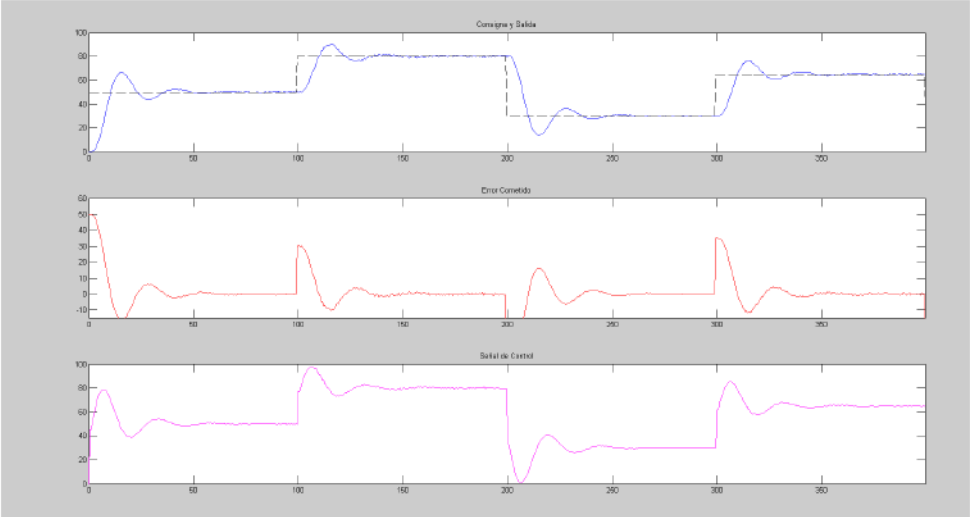


Figura 8.1.4.1 – Gráfica Ziegler-Nichols para un 40 % de punto de trabajo

- Ziegler-Nichols con Poca Sobreoscilación

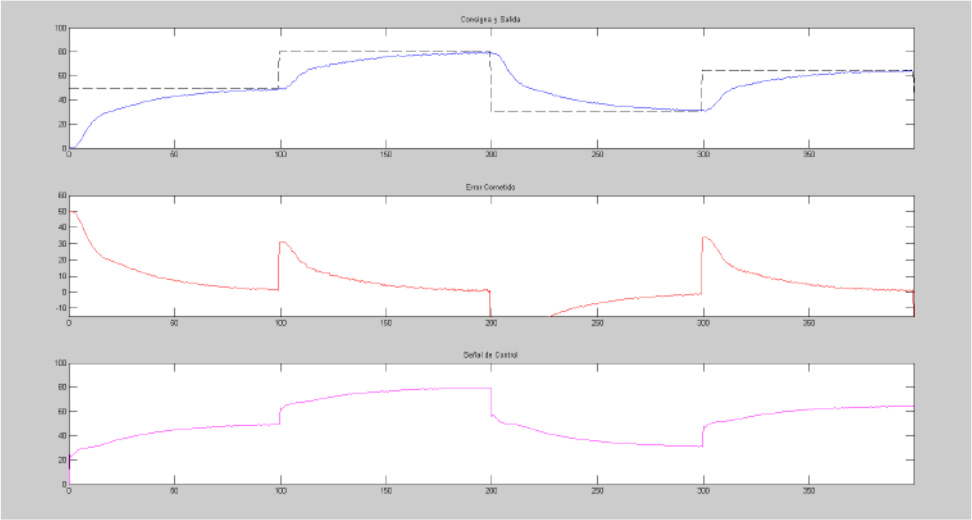


Figura 8.1.4.2 – Gráfica Ziegler-Nichols con Poca Sobreoscilación para un 40 % de punto de trabajo

- Ziegler-Nichols No Overshoot

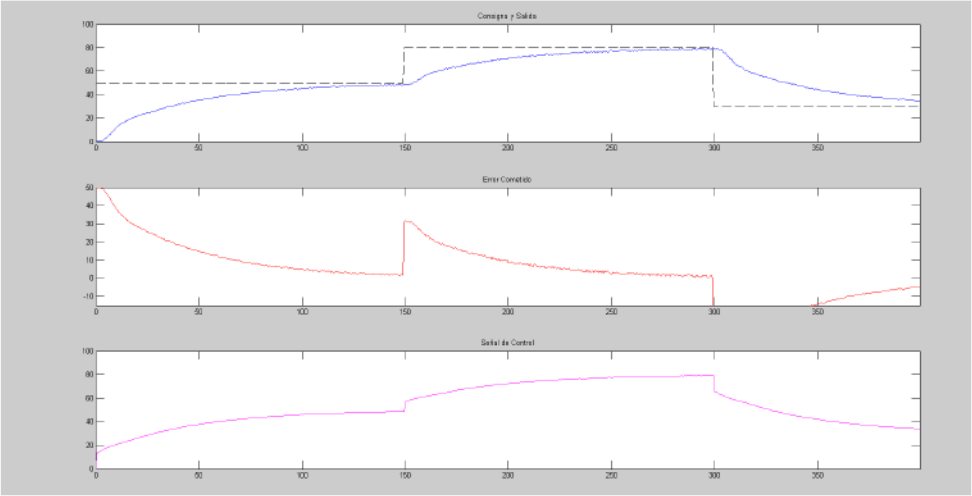


Figura 8.1.4.3 – Gráfica Ziegler-Nichols con No Overshoot para un 40 % de punto de trabajo

Si se realiza una comparación entre los tres métodos anteriores, ya que provienen todos del Ziegler-Nichols. Se pueden observar que en la figura 8.1.4.2 y 8.1.4.3 la sobreoscilación disminuye respecto de la figura 8.1.4.1.

Además en Ziegler-Nichols (figura 8.1.4.1), el tiempo de establecimiento del sistema es mayor y al reducir la sobreoscilación el sistema se hace más lento (figuras 8.1.4.2 y 8.1.4.3). Esto es debido a la constante proporcional K_p, ya que cuanto mayor sea su valor más rápido será el sistema.

En Ziegler-Nichols No Overshoot por tener un tiempo de establecimiento menor, fue necesario reducir la consigna para que el sistema llegase al régimen permanente en un tiempo razonable.

- Tyreus-Luyben

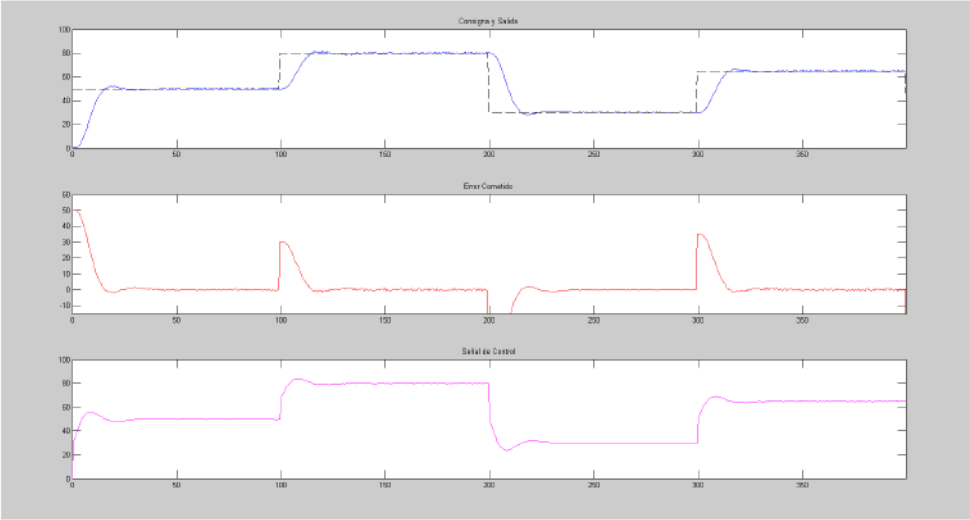


Figura 8.1.4.4 – Gráfica Tyreus-Luyben para un 40 % de punto de trabajo

El método de Tyreus-Luyben respeto al método Ziegler-Nichols mejora la sobreoscilación del sistema así como el tiempo de establecimiento y el sistema se estabiliza más rápido que en el anterior. Esto se puede observar en la figura 8.1.4.4

Como se puede ver en las tres figuras anteriores, 8.1.4.2, 8.1.4.3 y 8.1.4.4, se han utilizado diferentes puntos de consgina cada cierto número de muestras, en este caso cada 100, en la misma prueba. Además se observa que el error de posición se vuelve cero para conseguir la consigna deseada como es lógico al tratarse de reguladores PID.

8.2. Regulador PID Adaptativo

Como se ha mencionado anteriormente se ha realizado un PID adaptativo para la comparación de resultados con el PID estándar. Este tipo de regulador trabajará en tiempo real.

Para la implementación del regulador sobre la planta de control de nivel se debe partir de una función de transferencia genérica como la expuesta en la ecuación 7.3.2.2, ya que el primer paso ha de ser el de la obtención de una función de transferencia lo más fiel posible a la real de la planta, mediante el algoritmo RLS (explicado en la sección 7.3.2) para la identificación de la planta.

A continuación se mostrará el flujograma (figura 8.2.0.5) para la implementación de este regulador en el sistema real.

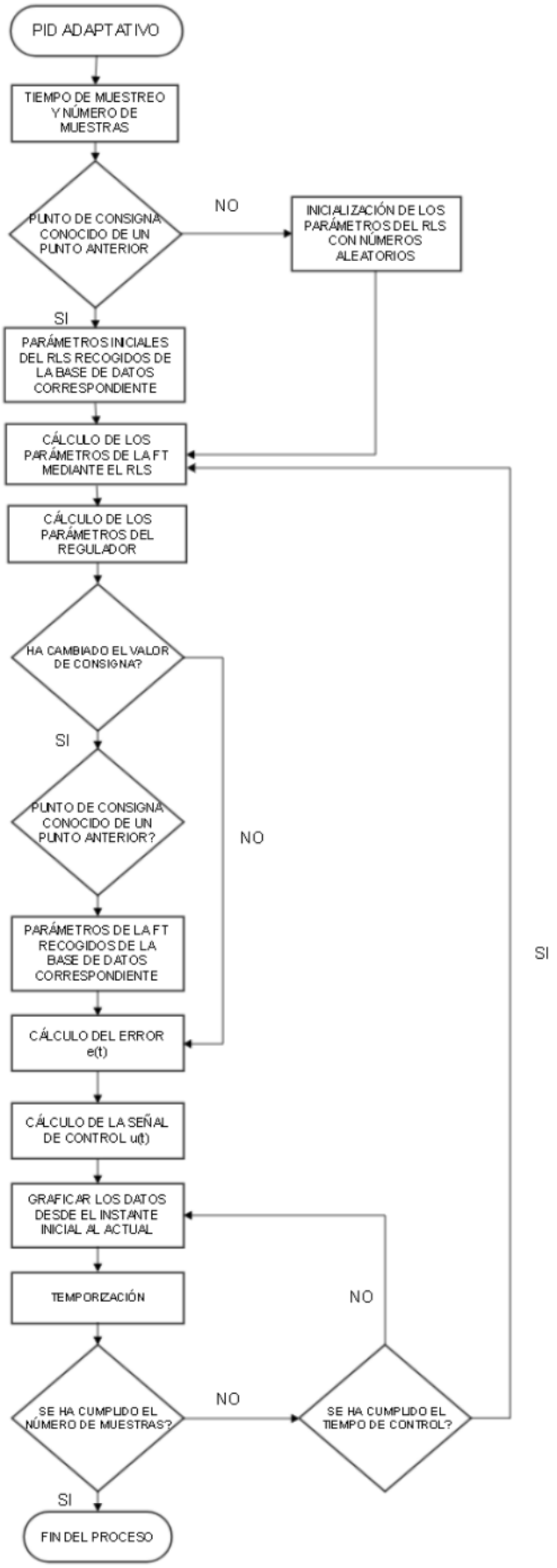


Figura 8.2.0.5 – Flujograma de la implementación del PID adaptativo

En el presente proyecto se ha optado por una función de transferencia de grado 1 y 2 de numerador y denominador respectivamente ya que la planta de control es un sistema de segundo orden, ya que se ha comprobado experimentalmente que la respuesta de la planta de control puede asimilarse a la de un sistema de segundo orden con retardo. obteniendo la siguiente función de transferencia equivalente:

$$\boxed{G(z) = \frac{b_1 z}{z^2 + a_1 z + a_0}} \tag{8.2.0.1}$$

Se puede observar en la ecuación 8.2.0.1, que se desprecia el término independiente del numerador, siendo la función completa la ecuación 7.3.2.4.

Una vez seleccionada la función de transferencia, ecuación 8.2.0.1, se procede a la implimentación del regulador siguiendo una serie de pasos para su correcto funcionameniento.

8.2.1. Inicialización de los parámetros de la Función de Transferencia a partir del RLS

Los parámetros $b1$, $a1$ y $a2$ de la ecuación 8.2.0.1 se deben obtener, como ya se ha explicado, mediante el algoritmo RLS. Para ello el primer paso a seguir es determinar un periodo de muestreo T y un tiempo de simulación t; siendo el número de muestras calculado en función de estos dos parámetros mediante el comando $floor$ de $MatLab$. Se calculará el número de muestras, siendo $muestras = floot(t/T)$.

Para proceder al cálculo del RLS se deben inicializar determinados parámetros, como se ha explicado en la sección 7.3.2, ya que deben ir realimentándose. Esta inicialización se ha realizado con números aleatorios mediante la función $rand$ de $MatLab$, pudiendo consultar su código en el anexo 13.4.2.

Ya que el periodo de muestreo no tiene porque coincidir con el usado para los cálculos de la identificación, es importante que el resultado de multiplicar el periodo de muestreo T por el número de muestras n_RLS (número de muestras que han de pasar para realizar la identificación, filtrado de la señal), coincida con el periodo de control (periodo al que se calcula los parametros del regulador y la señal de control).

En el presente proyecto se han usado diferentes periodos para la identificación y adquisición por dos motivos principales:

- Realizar un código que permita tener varios periodos internamente.
- Evitar ruido indeseado, promediándose las lecturas entre cada iteración de identificación.

8.2.2. Cálculo de los parámetros del regulador [2]

Calculados los parámetros de la función de transferencia, se tiene todo lo necesario para llevar a cabo el cálculo de los parámetros del regulador PID. Éste cálculo se ha realizado para un PID autoajustable basado en asignación de polos.

Este método parte de una función de transferencia como la dada por la ecuación 8.2.0.1 y un regulador PID de velocidad con la siguiente función de transferencia (ecuación 8.2.2.1)

$$\boxed{G_r(z^{-1}) = \frac{p_0 + p1z^{-1} + p2z^{-2}}{1 - z^{-1}}} \tag{8.2.2.1}$$

Y siendo el polinomio característico de la siguiente forma $1 + G_pG_r = 0$, se tiene:

$$\boxed{z^{-3} + \frac{a_1 - a_2 + b_1p_1}{b_1p_1 + a_2}z^{-2} + \frac{b_1p_0 - a_1 - 1}{b_1p_2 + a_2}z^{-1} + \frac{1}{b_1p_2a_2} = 0} \tag{8.2.2.2}$$

Se debe imponer que el sistema en bucle cerrado tenga un polo triple denominado a en el plano z^{-1}, por lo que su valor debe ser el de un número real de módulo mayor que uno. Aplicando este polo triple, el polinomio característico queda de la forma, ecuación 8.2.2.3.

$$\boxed{(z^{-1} - a)^3 = z^{-3} - 3az^{-2} + 3a^2z^{-1} - a^3} \tag{8.2.2.3}$$

Y al igualar los coeficientes de 8.2.2.1 y 8.2.2.3, se obtienen los parámetros de la FT del regulador:

$$p_0 = \frac{1}{b_1}(1 + a_1 - \frac{3}{a}) \tag{8.2.2.4}$$

$$p_1 = \frac{1}{b_1}(a_2 - a_1 + \frac{3}{a^2}) \tag{8.2.2.5}$$

$$p_2 = \frac{1}{b_1}(a_2 + \frac{1}{a^3}) \tag{8.2.2.6}$$

Dando lugar a las ganancias K_p (ganancia proporcional), K_i (ganancia integral) y K_d (ganancia derivativa), necesarias para el cálculo de la señal de control del regulador que se han visto en la sección 7.1.1

Los pasos a seguir para la obtención de estos parámetros son los siguientes y se pueden consultar en el anexo 13.4.3.

1. Se pasan a la función los valores de Numerador y Denominador de la función de transferencia.

2. Se asigna el valor del polo triple en el plano z^{-1} de módulo mayor que uno, en este caso se ha optado por a=10.

3. Se aplican las fórmulas 8.2.2.4, 8.2.2.5 y 8.2.2.6 para obtener los parámetros de la función de transferencia.

4. Una vez se obtienen esos parámetros se pasa al cálculo de las constantes del regulador K_p, K_i y K_d.

5. Cálculo de la señal de control del regulador.

8.2.3. Cálculo de la señal de control del regulador [i]

Por último se ha de calcular la señal de control del regulador, quinto punto de la enumeración anterior que será explicado en mayor profundidad ya que es un paso más complejo que los anteriores.

Este cálculo se realizará a partir de la ecuación en diferencias discretizada. El código de la función puede ser cosultado en la sección 13.4.4.

Los pasos a seguir para el cálculo de la señal de control se basan en:

1. Se deben pasar los valores de K_p, K_i y K_d calculados en el paso anterior así como la señal de control y el error en ese instante.

2. Basándose en la ecuación 7.1.1.1, se deben calcular los parámetros q_0, q_1 y q_2. Estos parámetros vienen dados por las siguientes ecuaciones, ya mostradas en 13.4.4.

$$q_0 = K_p + (1 + \frac{T}{2T_i} + \frac{T_d}{T}) \tag{8.2.3.1}$$

$$q_1 = K_p(\frac{T}{2T_i} - 1 - 2\frac{T_d}{T}) \tag{8.2.3.2}$$

$$q_2 = K_p\frac{T_d}{T} \tag{8.2.3.3}$$

Como se puede observar en las ecuaciones 8.2.3.4, 8.2.3.5 y 8.2.3.6 aparecen los parámetros T_i y T_d, que en este caso deben ser sustituidos por las ganancias proporcional y derivativa pudiendose ver la relación entre estos en la tabla 10.2.0.1. De esto modo las nuevas ecuaciones de q_0, q_1 y q_2 quedan del siguiente modo:

$$q_0 = K_p(\frac{K_iT}{2} + \frac{K_d}{T}) \tag{8.2.3.4}$$

$$q_1 = \frac{K_iT}{2} - K_p - 2\frac{K_d}{T} \tag{8.2.3.5}$$

$$q_2 = \frac{K_d}{T} \tag{8.2.3.6}$$

3. Calculados estos parámetros se aplica la ecuación 7.1.1.1 y se obtiene la nueva señal de control del regulador PID

   Como ya se ha comentado anteriormente, es importante limitar la señal, ya que se trabaja con una señal que va del 0 - 100 % de nivel de consigna, de este modo se realiza el efecto anti-windup eliminando la parte integral acumulada en exceso. Se puede consultar el código para limitar la señal en el anexo 13.4.4.

8.2.4. Resultados Obtenidos con el PID autoajustable

Una vez realizados todos los pasos anteriores se procede a la ejecución del código presentado en la sección 13.4.1 del que se obtendrán los resultados gráficamente para diferentes valores de consigna.

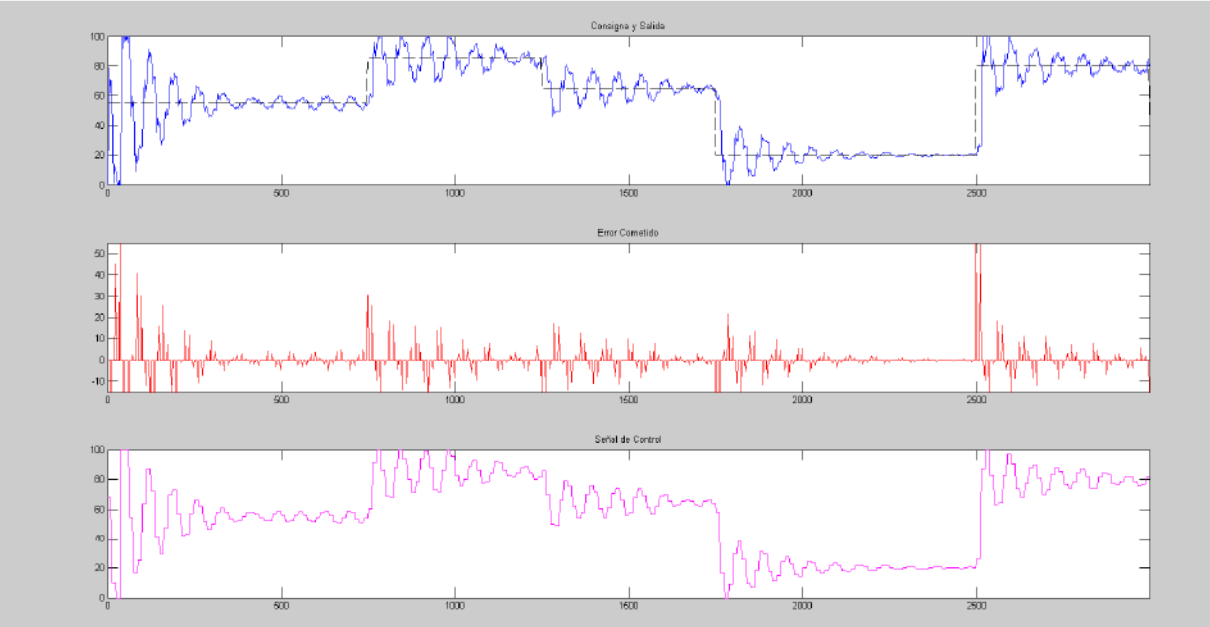


Figura 8.2.4.1 – Gráfica PID Adaptativo

En la figura 8.2.4.1, se parte de un nivel de consigna del 55 % y cada cierto número de muestras, el SP va cambiando. Se ha tomado un número de muestras de manera que para cada valor de consigna diferente, el regulador sea capaz de minimizar el error.

Al estar ante un sistema en el que los parámetros de la función de transferencia son inicializados con números aleatorios y calculados cada cierto número de muestras mediante el algoritmo RLS, los parámetros deben ser recalculados.

Se observa en la figura 8.2.4.1 que el tiempo de estabilización del sistema para el primer punto de consigna (55 %) es muy alto, ya que la FT del sistema está siendo identificada.

El tiempo de establecimiento para los puntos de consigna posteriores va a depender de los valores del numerador y denominador de la función de transferencia y del error obtenido en el SP anterior.

Si la diferencia entre el SP anterior y el actual es pequeña, el tiempo de establecimiento se reducirá, y si es grande se tendrá nuevamente un tiempo de establecimiento alto, ya que

se obtiene un mayor error, lo que conlleva que el sistema tarde más en estabilizarse. Esto se puede observar en la figura 8.2.4.1:

- 85 % a 65 % el tiempo de estabilización se reduce
- 55 % a 85 % el tiempo de estabilización es alto

Se puede observar en la figura 8.2.4.1 que la señal de control sigue a la señal de salida, siendo éste el objetivo buscado por el regulador PID trabajando en lazo cerrado, consiguiendo así un error del sistema prácticamente igual a cero.

Se dice que el error es prácticamente cero debido a las perturbaciones o ruidos a las que está sometida la planta de nivel, visualizándose por lo tanto, unas pequeñas oscilaciones sobre el punto de consigna a conseguir. Haciendo zoom en una parte de la gráfica 8.2.4.1 se puede observar en la figura 8.2.4.2 lo comentado.

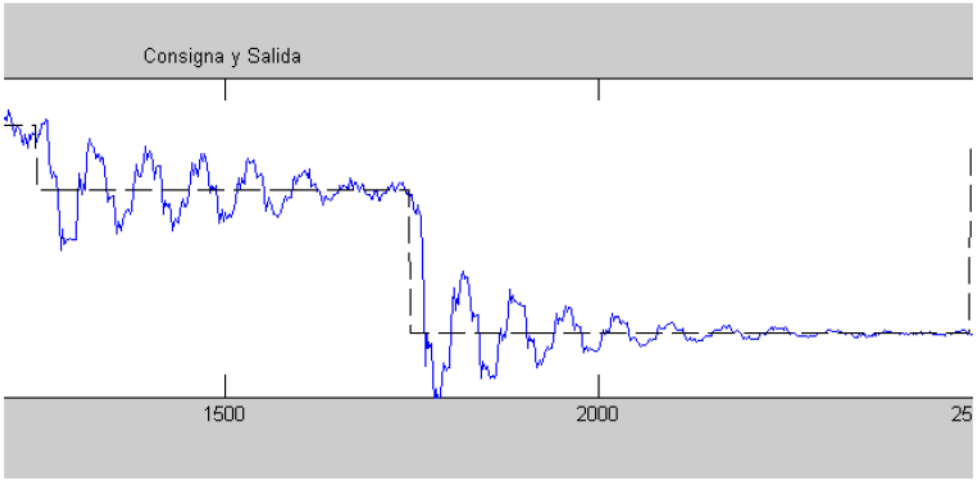


Figura 8.2.4.2 – Zoom gráfica 8.2.4.1 para visionado de ruido

8.2.4.1. Mejora del tiempo de estabecimiento

Para corregir el tiempo de estabilización de la identificación del sistema, se han generado una serie de ficheros para diferentes valores de consigna, en los que se han guardado los valores que se fueron obteniendo durante la identificación del numerador y denominador de la FT y del error. En este proyecto se ha optado por valores del 40 %, 50 %, 60 % y 80 %.

De este modo, si el valor de consigna es uno de los citados en el párrafo anterior, se usarán como punto de partida, los últimos valores calculados de los coeficientes del numerador y denominador. Evitando así dar valores aleatorios a estos parámetros, y garantizando un menor tiempo de establecimiento, ya que se parte de valores conocidos del sistema. Ésto ha sido realizado mediante una función y su código puede ser consultado en la sección 13.4.5

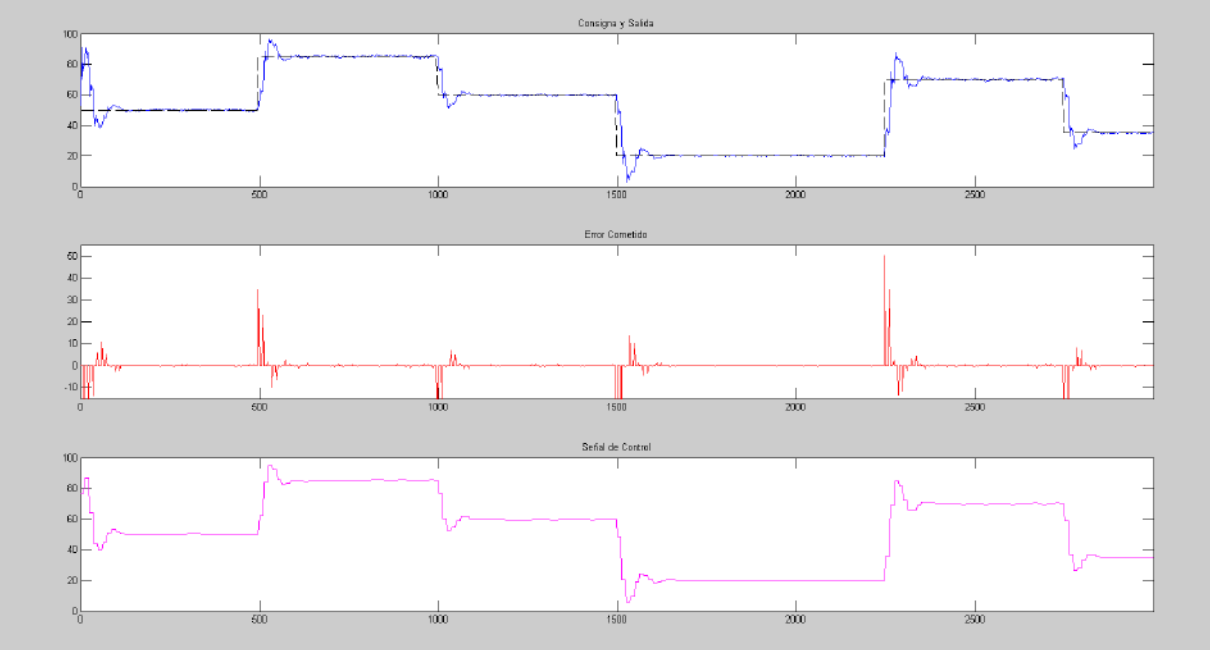
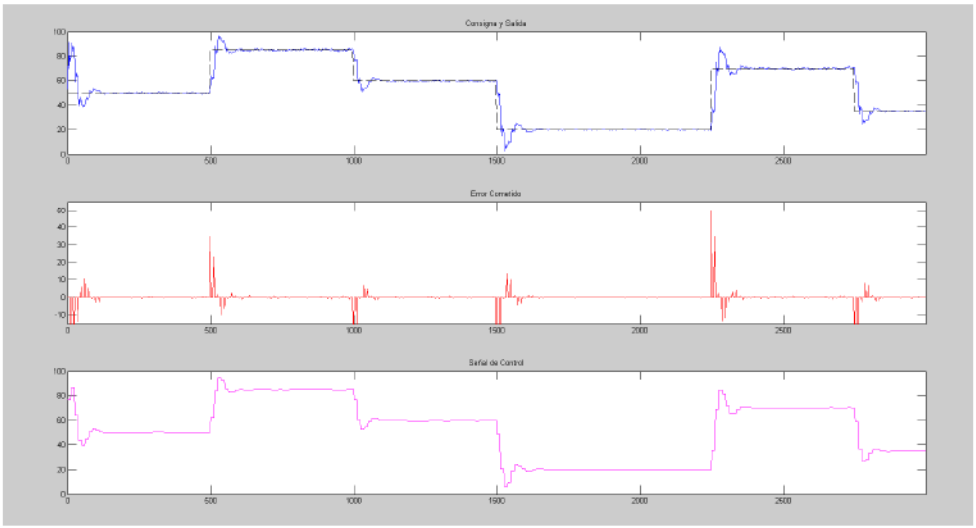
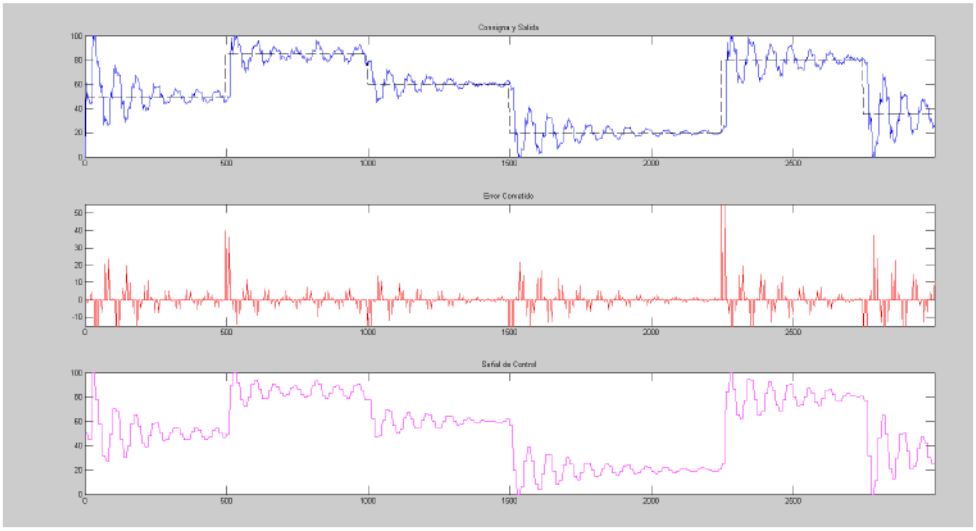


Figura 8.2.4.3 – PID Adaptativo con tiempo de estabilización reducido

En la figura 8.2.4.3, se observa que el primer valor de consigna es del 50 % por lo que los datos iniciales serían recogidos del fichero, como ya se ha explicado, y reduciendo así el tiempo de establecimiento.



(a) PID recogiendo variables del fichero



(b) PID inicializado con valores aleatorios

Figura 8.2.4.4 – Gráficas PID Adaptativo

En la figura 8.2.4.4, se puede realizar una comparativa para los mismos puntos de trabajo. En la gráfica 8.4(b) se puede apreciar que al tomar valores aleatorios incialmente el tiempo de establecimiento es considerablemente alto, mientras que para la gráfica 8.4(a), para la que se han recogido los datos del fichero, este tiempo se reduce haciendo que el sistema alcance de forma más rápida un error próximo a cero.

Esta recogida de los datos desde el fichero, se debe realizar siempre que se cambie el valor de consigna (cada cierto número de muestras) y éste sea un valor conocido para que el existe una base de datos.

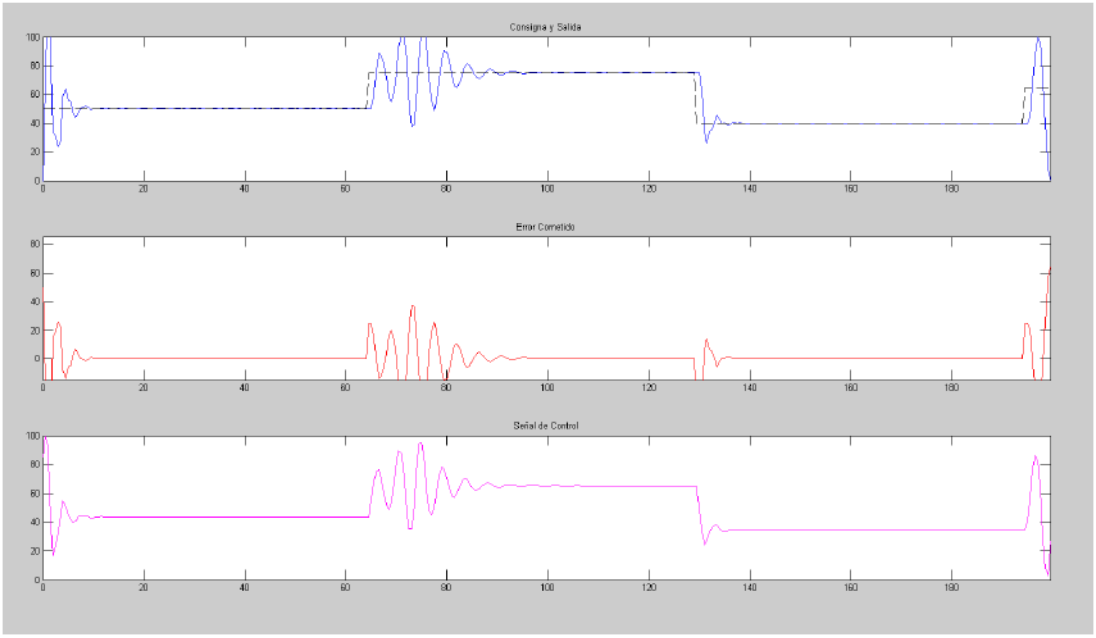


Figura 8.2.4.5 – Gráfica PID Adaptativo con diferentes valores de SP

En la figura 8.2.4.5, se observa lo antes comentado; para el SP inicial 50 % el tiempo de establecimiento es bajo ya que se están recogiendo los datos del fichero creado.

Para el punto de consigna posterior (85 %), los datos no han sido recogidos de un fichero por lo que el tiempo de establecimiento para ese SP vuelve a aumentar y para el siguiente punto visible en la gráfica ocurre lo mismo que para el 50 % (en este caso un SP=60 %).

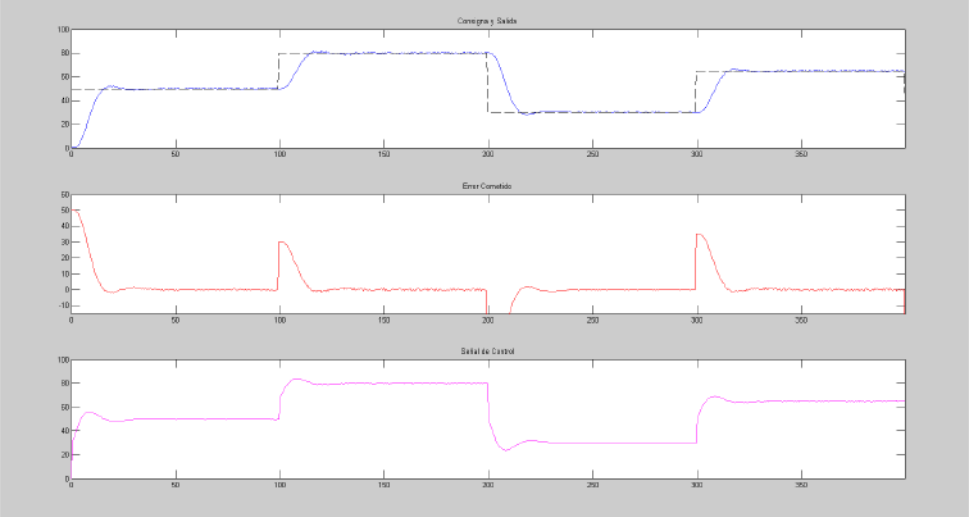
8.3. Comparación de los resultados obtenidos en el PID Estándar y PID Adaptativo

Estudiados los dos reguladores implementados se pueden apreciar ciertas diferencias entre ellos.

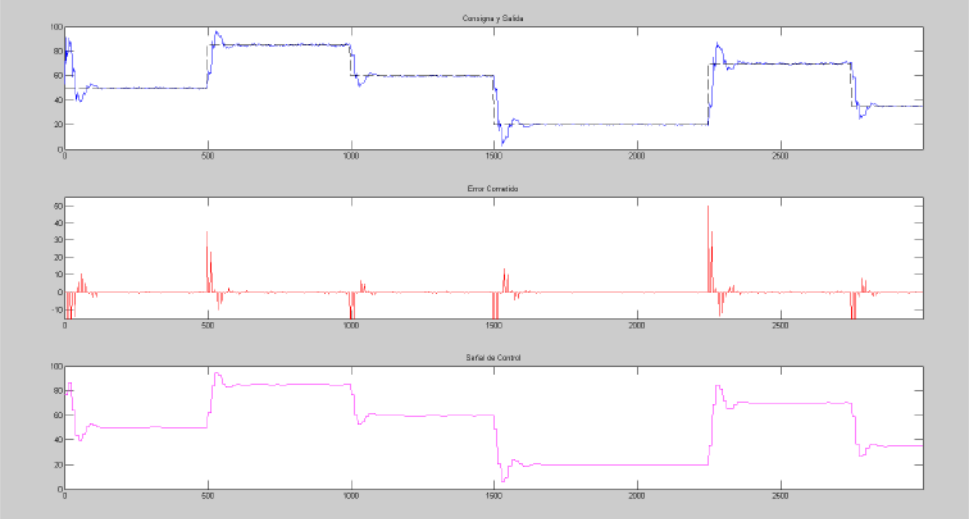
Para el PID estándar se debe partir de datos conocidos, por lo que se somete al sistema a unas especificaciones concretas, en este caso se partía de los resultados obtenidos en el Método del Relay-Feedback para un punto de operación determinado.

Por lo contrario, para el PID adaptitivo no es necesario conocer ningún dato acerca del sistema, ya que la identificación de su función de transferencia se realiza en tiempo real. Ésto ahorra tener que realizar cálculos previos y el sistema que está siendo identificado se aproxima más al sistema real. Además éste es capaz de adaptarse a distintas plantas y cambios en las mismas debido a envejecimiento o deterioro de componentes por su uso.

Una desventaja que podría tener el PID Adaptativo frente al PID estándar es que es un tipo de regulador más sensible al ruido, ya que como antes se comentaba los parámetros están siendo calculados en función de cómo reacciona el sistema.



(a) PID estándar Tyreus Luyben



(b) PID Adaptativo

Figura 8.3.0.6 – Gráficas PID Adaptativo

En la figura 8.3.0.6 se puede apreciar una simulación del PID estándar mediante Tyreus Luyben, siendo éste el método que proporciona la mejor respuesta para el sistema basado en el PID estándar, figura 8.6(a), y del PID adaptativo, figura 8.6(b). Como se puede ver en la figura 8.6(b), se aprecia que este tipo de PID es más sensible al ruido al contrario del estándar, figura 8.3.0.6.

En la figura **??**, apenas existe sobreoscilación debido al método empleado para su implementación y los cálculos previos. Por lo contrario en la figura **??**, hay mayor sobreoscilación por lo antes comentado; se realiza la implementación en tiempo real, por lo que el sistema tarda más en estabilizarse.

TÍTULO: **IMPLEMENTACIÓN DE UN ALGORITMO DE CONTROL ADAPTATIVO**

ANEXOS

PETICIONARIO: **ESCUELA UNIVERSITARIA POLITÉCNICA**

AVDA. 19 DE FEBREIRO, S/N

15405 - FERROL

FECHA: JUNIO DE 2019

AUTOR: EL ALUMNO

Fdo.: MARÍA REFOJO FERREIRÓS

Índice del documento ANEXOS

9 Documentación de partida

Escuela Universitaria Politécnica Ferrol
UNIVERSIDADE DA CORUÑA

Intranet eup

ESCUELA UNIVERSITARIA POLITÉCNICA

ASIGNACIÓN DE TRABAJO FIN DE GRADO

En virtud de la solicitud efectuada por:
En virtud da solicitude efectuada por:

APELLIDOS, NOMBRE: *Refojo Ferreirós,María*
APELIDOS E NOME:

DNI: [redacted] **Fecha de Solicitud:** Feb2018
DNI: *Fecha de Solicitude:*

Alumno de esta escuela en la titulación de Grado en Ingeniería en Electrónica Industrial y Automática, se le comunica que la Comisión de Proyectos ha decidido asignarle el siguiente Trabajo Fin de Grado:

O alumno de esta escola na titulación de Grado en Enxeñería en Electrónica Industrial e Automática, comunícaselle que a Comisión de Proxectos ha decidido asignarlle o seguinte Traballo Fin de Grado:

Título T.F.G:Implementación de un algoritmo de control adaptativo

Número TFG: *770G01A156*

TUTOR:(Titor) *Casteleiro Roca,José Luis*

COTUTOR/CODIRECTOR: *Héctor Quintián Pardo*

La descripción y objetivos del Trabajo son los que figuran en el reverso de este documento:

A descrición e obxectivos do proxecto son os que figuran no reverso deste documento.

Ferrol a Martes, 27 de Febrero del 2018

Retirei o meu Traballo Fin de Grado o dia _____ de _____ do ano _____

Fdo: Refojo Ferreirós,María

DESCRIPCIÓN Y OBJETIVO:Objeto:
Este Trabajo Final de Grado abordará la implementación de un algoritmo de control adaptativo basado en el modelo de planta. Se implementara inicialmente un algoritmo de identificación RLS, que proporcione una función de transferencia con una serie de especificaciones concretas. Posteriormente, se implementará un regulador PID, que usará las variables de la función de transferencia identificada para adaptar sus ganancias internas en cada momento.

Alcance:
- Estudio del algoritmo RLS para identificación on-line.
- Creación de las funciones necesarias para ejecutar el algoritmo de identificación on-line con diferentes configuraciones de la función de transferencia.
- Realización de las pruebas del correcto funcionamiento de las funciones en simulación (con funciones de transferencia conocidas).
- Creación de una "base de datos" con las funciones de transferencia identificadas en función del punto de trabajo, para partir de un estado conocido para próximas identificaciones.
- Implementación de un regulador adaptativo basado en el modelo de planta para controlar una planta de laboratorio.
- Comparación de los resultados usando el regulador adaptativo y un regulador PID estándar (sintonizado para un punto de trabajo concreto).

10 Sistema de control de procesos

Como ya se ha mencionado, la Ingeniería de Control se centra en el control de procesos y ha ido mejorando a lo largo de los años optimizando los procesos a controlar, tanto de manera económica como técnica, así como las condiciones de los operarios.

El control en la industria pretende obtener el resultado deseado en el proceso sin una intervención humana de manera directa y minimizándola, a través de sistemas SCADA, controladores lógicos programables, etc.

Un ejemplo de un campo de aplicación del control industrial es el que se realiza en el presente proyecto, donde se realiza el control del nivel de agua de un depósito de la planta del "Laboratorio de Optimización y Control"(figura 3.1.0.2). Además de procesos industriales, los sistemas automáticos de control también juegan un papel importante en los hogares, mejorando la calidad de vida con aparatos electrónicos como por ejemplo desde una lavadora hasta la domótica de un edificio.

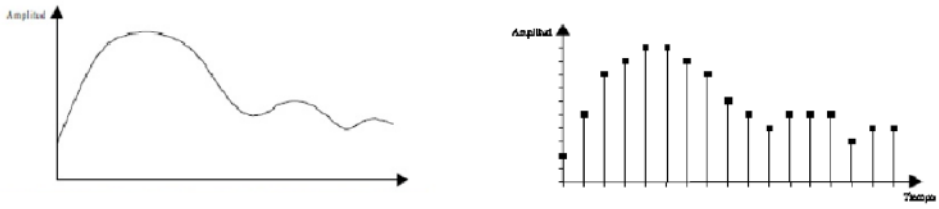
La primera aparición de la ingeniería de control fue a través del control analógico, el cual sigue siendo utilizado. Este tipo de control se basa en el dominio del tiempo continuo, donde la base de cálculo es la transformada de $Laplace$.

En la Ingeniería de Control existen procesos que son controlados sólo en función de las señales de entrada al sistema (sistemas en cadena abierta), y aquellos que además de las señales de entrada precisan de las salidas del sistema, es decir, son realimentadas (sistemas en cadena cerrada).

10.1. Control discreto

La aparición del control digital, ha dado lugar al control discreto, en el que la base de cálculo es la transformada Z.

En este tipo de control, a diferencia del control continuo, las variables de entrada y salida son medidas cada cierto tiempo con un intervalo de tiempo constante, es decir, son muestreadas. Por ello aparece una nueva variable, el tiempo de muestreo, de la cual dependerá la respuesta de proceso a controlar. En la figura 10.1.0.1, se puede observar la diferencia entre una señal continua (10.1(a)) y una señal discreta (10.1(b)), donde se puede observar el tiempo de muestreo del que se hablaba.



(a) Señal Continua (b) Señal Discreta

Figura 10.1.0.1 – Señales de Control

Para determinar la estabilidad en un sistema discreto existen diferentes técnicas como:

- Criterio de Jury: donde se analiza la estabilidad del sistema sin obtener las raíces de la ecuación característica.

- Criterio de Routh extendido: Se aplica el criterio de Routh de sistemas continuos a sistemas discretos.

Aunque el método más sencillo es determinarla a través de la posición de los polos del sistema en el plano Z, que se corresponden con las raíces de la ecuación característica.

Esta ecuación característica (10.1.0.2) se obtiene a partir de la función de transferencia (10.1.0.1) que se puede obetener a partir del diagrama de bloques del sistema, figura 10.1.0.2

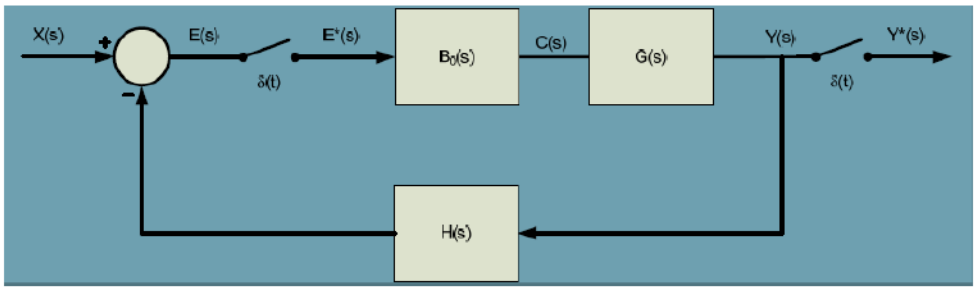


Figura 10.1.0.2 – Diagrama de Bloques

Siendo la función de transferencia:

$$\boxed{\frac{Y(z)}{X(z)} = \frac{B_0 G(z)}{1 + B_0 HG(z)}} \tag{10.1.0.1}$$

Y la ecuación característica que se deduce de (10.1.0.1)

$$\boxed{P(z) = 1 + B_0 HG(z)} \tag{10.1.0.2}$$

Es importante destacar que la ubicación de los polos del sistema (en cadena cerrada) o raíces del polinomio característico determinarán no solo la estabilidad del sistema (todos los polos dentro del circulo unidad en el plano Z), sino también el comportamiento de dicho sistema. Entre los ascpectos más importantes a tener en cuenta cabe destacar los siguientes:

- El sistema es estable si todos los polos, raíces de la ecuación característica, están dentro del circulo unidad.
- Los ceros, raíces del denominador de la función de transferencia, no afectan a la estabilidad.
- Un único polo en z=1 hace el sistema inestable.
- Un único polo en z=-1 hace el sistema oscilante.
- Un par de polos complejos conjugados sobre el circulo unidad hacen el sistema oscilante.
- Un polo múltiple sobre el circulo unidad, lo hace inestable.

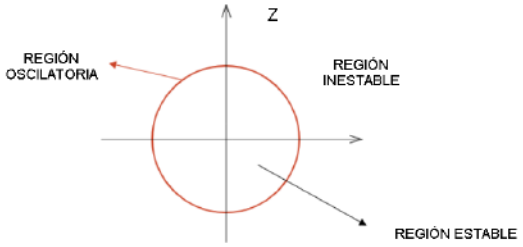


Figura 10.1.0.3 – Plano Z

La posición en la que se encuentren los polos y ceros en el sistema de control dependea también del periodo de muestreo, por lo que la estabilidad del sistema estará condicionada por su elección, siendo éste un parámetros crítico en los sistemas de control discreto. Para ello debe cumplirse en primera instan cia el teorema de Niquist, el cual determina que la frecuencia de muestreo ($T_s = 1/fs$) debe ser menor que dos veces la frecuencia máxima del sistema. En la práctica existen reglas empíricas que determinan que el periodo de muestreo se puede establecer como $1/3$ del tiempo de respuesta del sistema

10.2. Regulador PID [ii]

Antes de implementar un PID se deben conecer dos tipos de sistemas, el de lazo abierto (sin realimentación de la salida) y lazo cerrado (con realimentación de la salida).

En lazo abierto, la señal de salida no influye sobre la de entrada ya que no hay realimentación y por tanto no es capaz de actuar ante perturbaciones (figura 10.2.0.4).

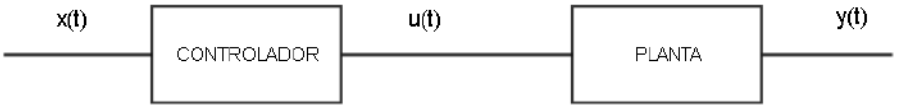


Figura 10.2.0.4 – Diagrama de Bloques de un Sistema de Lazo Abierto

En lazo cerrado, la salida influye sobre la entrada ya que hay realimentación. Es decir, ajustan la acción de control hasta obtener la salida deseada que se ha aplicado a la entrada (set point). De este modo actúa ante las perturbaciones que se produzcan en el sistema hasta conseguir el mínimo error (diferencia entre el set point y la señal realimentada).

Como ya se ha comentado anteriormente el regulador PID es un mecanismo de control de lazo cerrado que permite regular parámetros como velocidad, caudal, presión, etc. Su función es la de calcular la señal de control idónea (a partir del error que se ha medido) en cada momento con la finalidad de llevar la variable del proceso, la salida, al valor deseado.

Existen tres parámetros fundamentales en el algoritmo de control de un PID: Ganancia Proporcional (P), Tiempo Integral (I) y Tiempo Derivativo (D).

Es importante programarlos de manera adecuada para conseguir el objetivo del regulador, que el parámetro deseado siga al SP con un margen de error prácticamente nulo.

A continuación se puede observar el diagrama de bloques de un sistema de control en el que interviene un controlador PID, figura 10.2.0.5, donde el bloque PID es el regulador implementado al sistema de control, el actuador podría ser la parte de la DAQ que hace de D/A para llevar los datos a la planta junto con la bomba en este caso, el bloque planta es el sistema real (en este caso la planta de nivel del laboratorio de Optimización y Control) y el captador es nuevamente el DAQ donde ahora trabaja como A/D recogiendo los datos del sistema real a través del sensor de nivel descrito anteriormente.

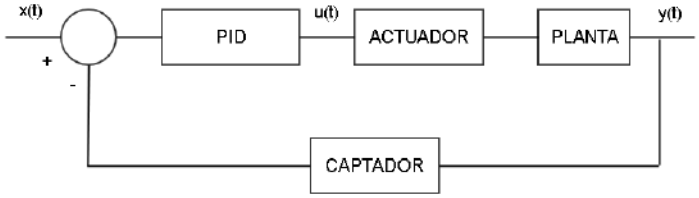


Figura 10.2.0.5 – Diagrama de Bloques con regulador PID

Se ha implementado este tipo de regulador ya que estamos ante un comportamiento no ideal por ser un sistema real. Estos tipos de sistemas están sometidos a perturbaciones y

presentan errores y el regulador PID ofrece unos buenos resultados por implementar las tres acciones ya mencionadas anteriormente, aunque cabe decir que un regulador será más o menos óptimo en función de las características del sistema al que es implementado.

La expresión de éste regulador en el dominio del tiempo viene determinada por la ecuación 10.2.0.3

$$u(t) = K_p \left(e(t) + \frac{1}{T_i} \int_0^t e(t) \cdot dt + T_d \cdot \frac{de(t)}{dt} \right) \tag{10.2.0.3}$$

Siendo:

- $u(t)$: la entrada de control aplicada a la planta en el instante t
- $e(t)$: el error en el instante t, siendo la diferencia entre la salida (o variable del proceso) y el punto de consigna.

Cabe destacar que los parámetros T_i y T_d que aparecen en la ecuación 10.2.0.3 podrán aparecer también como las ganancias K_i y K_d como ya se ha mencionado en las secciones 3.3.1.3 y 3.3.1.3. Esto dependerá de si se está ante un regulador PID estándar o un regulador no-estándar.

En el presente proyecto se tienen ambos casos, en el PID estándar se obtienen el tiempo integral (T_i) y el tiempo derivativo (T_d), a partir de los métodos empleados para la obtención de los parámetros del regulador (ajuste empírico).

Por lo contrario, para un regulador no-estándar, en este caso el PID adaptativo, al tratarse de un sistema implementado en tiempo real, se obtienen las gananancias, K_i y K_d, a través de los parámetros de la función de transferencia del regulador.

Estos parámetros están relacionados del siguiente modo, y en ambos casos, tanto en el PID estándar como en el adaptativo, se pueden aplicar estas relaciones, tabla 10.2.0.1

Parámetros	
Kp	K
Ki	Kp/Ti
Kd	Kp · Td

Tabla 10.2.0.1 – Parámetros Regulador PID

Para poder implementar el PID en tiempo discreto se debe partir del regulador en tiempo continuo que viene determinado por la ecuación 10.2.0.3. Ésta discretización es una aproximación del original.

11 Cálculos

11.1. Cálculos para el PID Estándar

En esta sección se irán mostrando los cálculos realizados para la implementación del PID estándar.

11.1.1. Valores parámetros Relay-Feedback

A continuación, en la tabla 11.1.1.1,se muestran los valores medios obtenidos mediante el método Relay-Feedback para diferentes puntos de trabajo de la planta de control. Como ya se ha comentado el cálculo de K_c se ha realizado según la ecuación 7.2.1.1

	40 %			50 %			60 %			80 %		
	Tc	**a**	**Kc**	**Tc**	**a**	**Kc**	**Tc**	**a**	**Kc**	**Tc**	**a**	**Kc**
	33	50.04	1.28	32	49.5	1.29	32	48.24	1.33	36	47.19	1.36
	31	49.08	1.30	33	49.52	1.29	32	49.03	1.31	34	47.6	1.34
	31	52.8	1.21	33	48.9	1.31	32	48.67	1.31	34	47.46	1.35
Media	**31.67**	**50.62**	**1.265**	**32.67**	**49**	**1.298**	**32.00**	**48.65**	**1.316**	**34.667**	**47.42**	**1.350**

Tabla 11.1.1.1 – Valores de 'Tc', 'Kc' y 'a' en función del punto de trabajo

11.1.2. Valores de los parámetros del PID

Dependiendo del modo de trabajo elegido se obtendrán diferentes valores para los parámetros K_p, T_i y T_d, en los que también va a influir el punto de trabajo elegido. Para los diferentes puntos de trabajo mostrados en la tabla 11.1.1.1 y siguiendo las ecuaciones para cada modo de trabajo explicados en la sección 7.2.2 se obtienen los parámetros mostrados en la tablas 11.1.2.1, 11.1.2.2, 11.1.2.3 y 11.1.2.4.

	PUNTO Y MODO TRABAJO	40 %			
		ZN	ZN Poca Sob	ZN No Over	TL
PARÁMETROS PID	Kp	0.76	0.42	0.25	0.57
	Ti	15.79	15.79	31.57	69.45
	Td	3.95	10.52	10.52	5.01

Tabla 11.1.2.1 – Valores de los parámetros del regulador para el 40 % de SP

	PUNTO Y MODO TRABAJO	50 %			
		ZN	ZN Poca Sob	ZN No Over	TL
PARÁMETROS PID	Kp	0.78	0.43	0.26	0.58
	Ti	16.34	16.34	32.6	71.87
	Td	4.08	10.89	10.89	5.19

Tabla 11.1.2.2 – Valores de los parámetros del regulador para el 50 % de SP

	PUNTO Y MODO TRABAJO	60 %			
		ZN	ZN Poca Sob	ZN No Over	TL
PARÁMETROS PID	Kp	0.79	0.43	0.26	0.59
	Ti	16.00	16.00	32.00	70.40
	Td	4.00	10.67	10.67	5.08

Tabla 11.1.2.3 – Valores de los parámetros del regulador para el 60 % de SP

	PUNTO Y MODO TRABAJO	80 %			
		ZN	ZN Poca Sob	ZN No Over	TL
PARÁMETROS PID	Kp	0.81	0.45	0.27	0.61
	Ti	17.33	17.33	34.67	76.27
	Td	4.33	11.56	11.56	5.50

Tabla 11.1.2.4 – Valores de los parámetros del regulador para el 80 % de SP

11.2. Cálculos para el PID Adaptativo

Los cálculos para la obtención de los parámetros de la función de transferencia se realizan como se ha mostrado en la sección 7.3.2 y una vez obtenidos éstos se pasaba al cálculo de los parámetros del regulador.

Para calcular los parámetros del regulador se debía imponer un polo triple en la zona de inestabilidad del sistema, tomándose un valor de $a = 10$.

Para explicar esto con un ejemplo, si se obtuvieron los siguiente valores para los parámetros de la función de transferencia del sistema para un punto de trabajo del 50 %:

- $b_1 = 0{,}7772$
- $a_1 = -0{,}2083$
- $a_2 = -0{,}0133$

Aplicando las fórmulas 8.2.2.4, 8.2.2.5 y 8.2.2.6 se obtienen los parámetros de la FT del regulador:

- $p_0 = 0{,}6327$
- $p_1 = 0{,}2895$
- $p_2 = 0{,}0158$

Y a partir de éstos y las fórmulas 7.1.1.5, 7.1.1.6 y 7.1.1.7 se calculan las constantes del regulador necesarias para el cálculo de la señal de control:

- $K_p = -0{,}3212$
- $K_i = 1{,}8760$
- $K_d = 0{,}0079$

Una vez obtenidas las constantes del regulador, el último paso, como ya se ha comentado, sería el cálculo de la señal de control utilizando para ello el regulador discretizado por $Tustin$.

Como ya se ha comentado en la sección 8.2.3, para el cálculo de la señal de control es necesario calcular una seria de parámetros; y en base a los resultados antes mostrados y a través de las ecuaciones 7.1.1.2, 7.1.1.3 y 7.1.1.4 se obtiene:

- $q_0 = 0{,}3971$
- $q_1 = 0{,}1225$
- $q_2 = -0{,}0837$

12 Resultados obtenidos para ambos reguladores

12.1. Resultados del PID estandar para todos los casos propuestos

1. **Resultados para 40 % de punto de trabajo**

   - Relay-Feedback

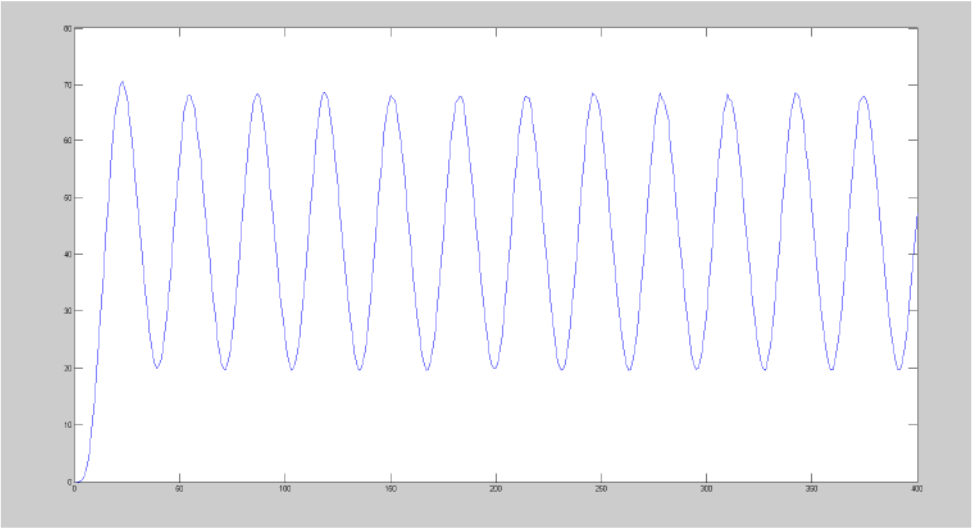


Figura 12.1.0.1 – Gráfica Relay-Feedback para punto de trabajo 40 %

- Ziegler-Nichols

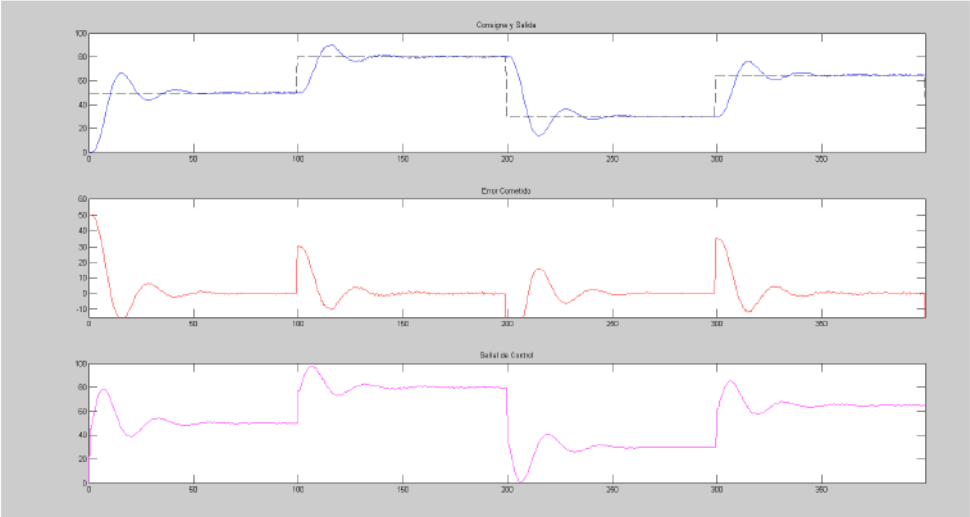


Figura 12.1.0.2 – Gráfica Ziegler-Nichols para un 40 % de punto de trabajo

- Ziegler-Nichols con Poca Sobreoscilación

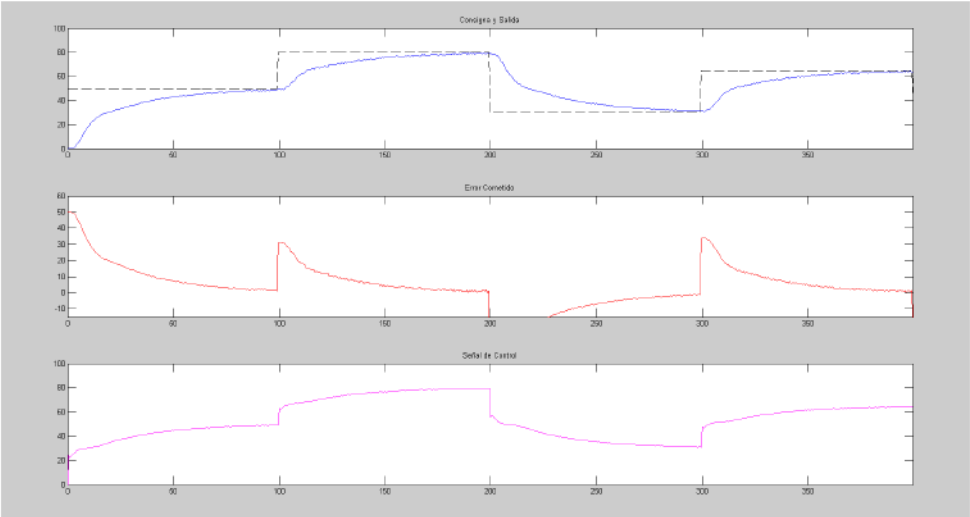


Figura 12.1.0.3 – Gráfica Ziegler-Nichols con Poca Sobreoscilación para un 40 % de punto de trabajo

- Ziegler-Nichols No Overshoot

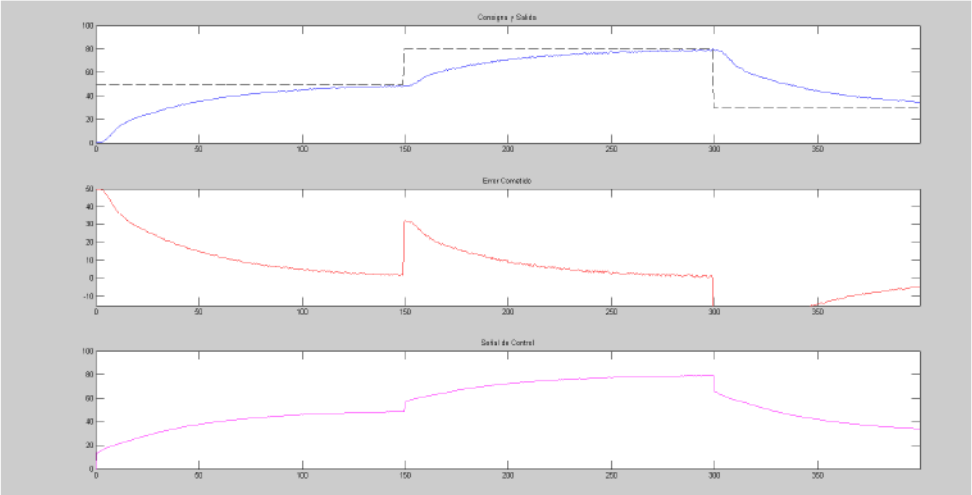


Figura 12.1.0.4 – Gráfica Ziegler-Nichols con No Overshoot para un 40 % de punto de trabajo

- Tyreus Luyben

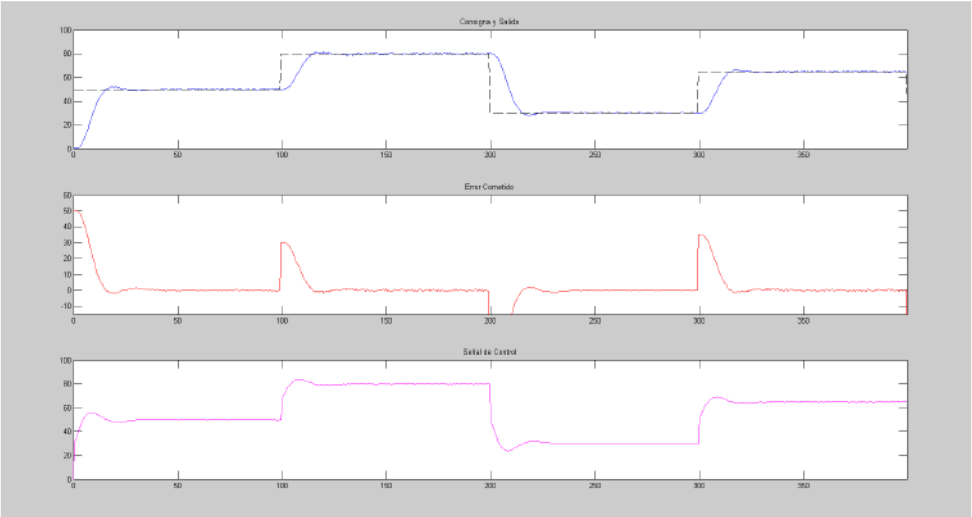


Figura 12.1.0.5 – Gráfica Tyreus-Luyben para un 40 % de punto de trabajo

2. Resultados para 50 % de punto de trabajo

- Relay-Feedback

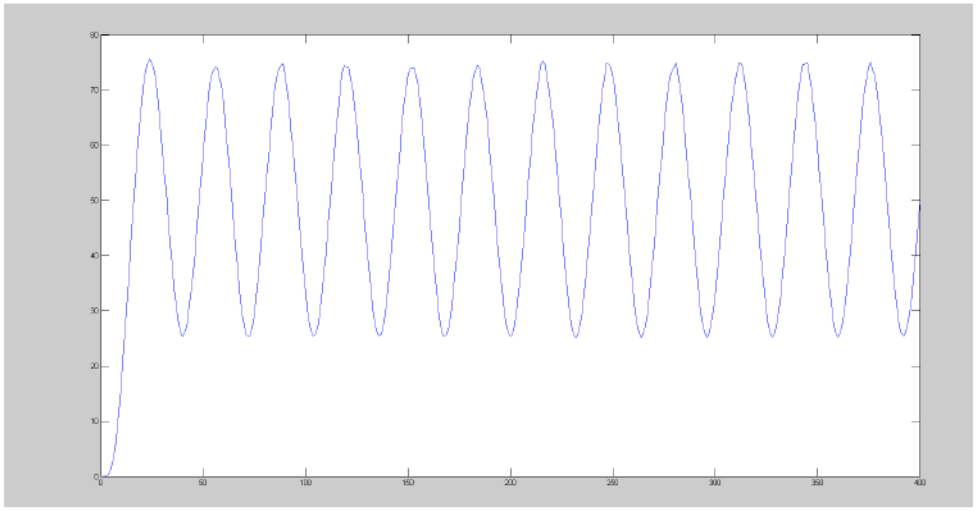


Figura 12.1.0.6 – Gráfica Relay-Feedback para punto de trabajo 50 %

- Ziegler-Nichols

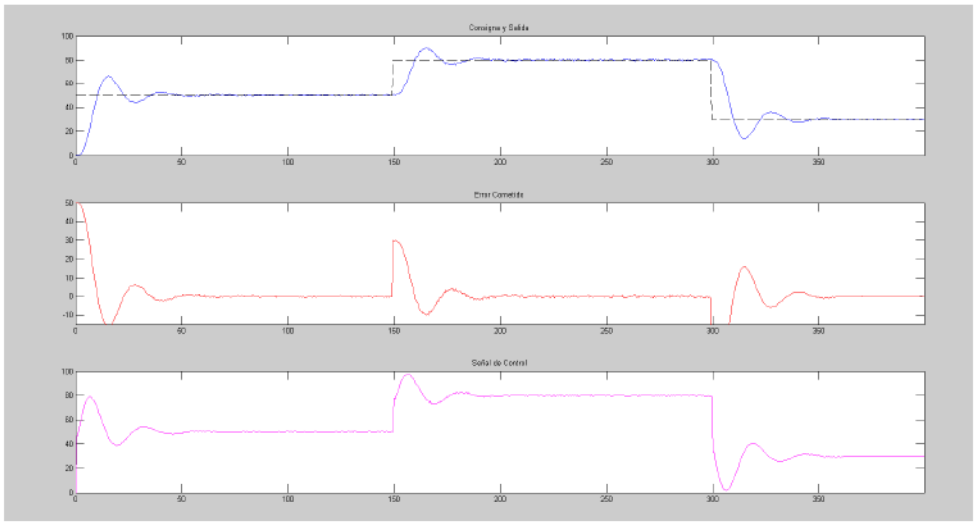


Figura 12.1.0.7 – Gráfica Ziegler-Nichols para un 50 % de punto de trabajo

- Ziegler-Nichols con Poca Sobreoscilación

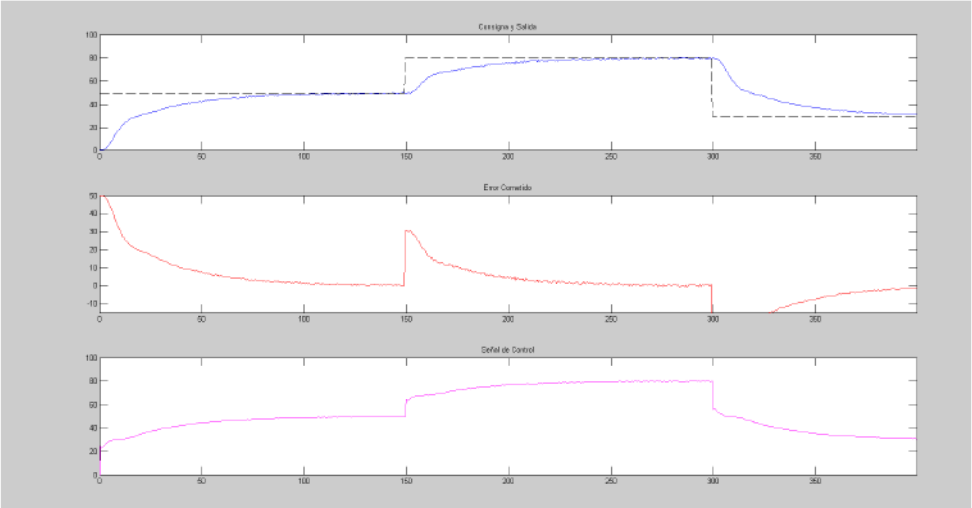


Figura 12.1.0.8 – Gráfica Ziegler-Nichols con Poca Sobreoscilación para un 50 % de punto de trabajo

- Ziegler-Nichols No Overshoot

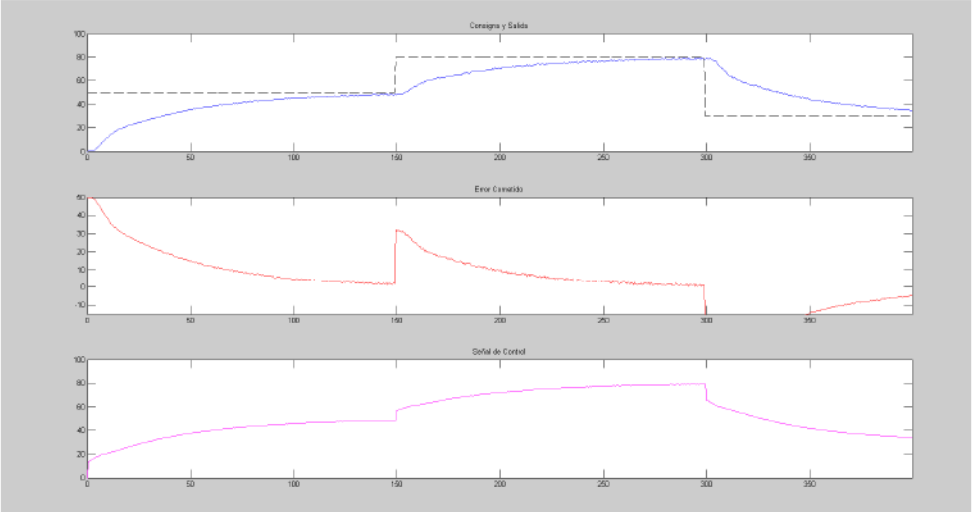


Figura 12.1.0.9 – Gráfica Ziegler-Nichols con No Overshoot para un 50 % de punto de trabajo

- Tyreus Luyben

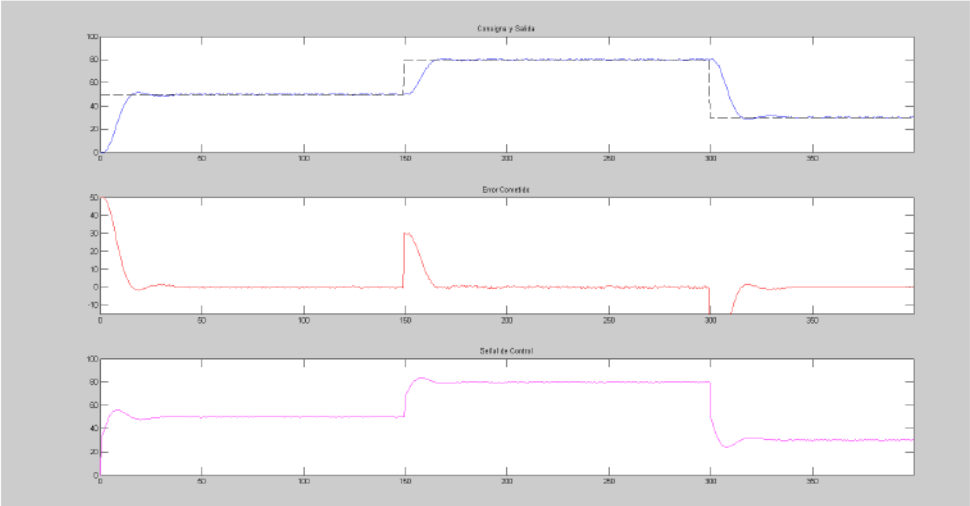


Figura 12.1.0.10 – Gráfica Tyreus-Luyben para un 50 % de punto de trabajo

3. **Resultados para 60 % de punto de trabajo**

- Relay-Feedback

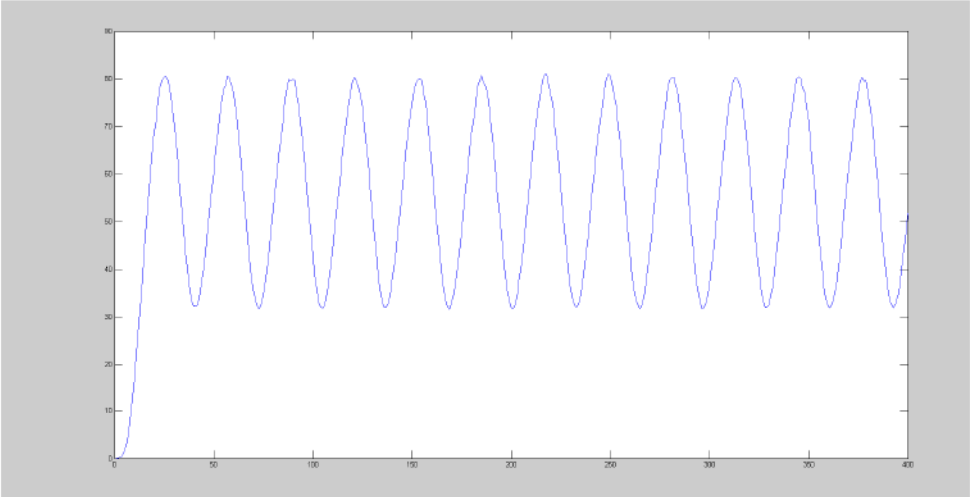


Figura 12.1.0.11 – Gráfica Relay-Feedback para punto de trabajo 60 %

- Ziegler-Nichols

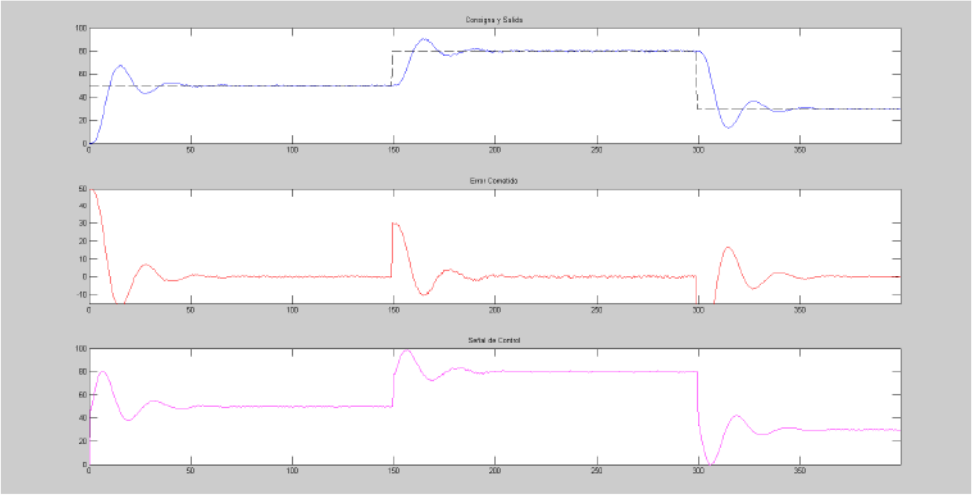


Figura 12.1.0.12 – Gráfica Ziegler-Nichols para un 60 % de punto de trabajo

- Ziegler-Nichols con Poca Sobreoscilación

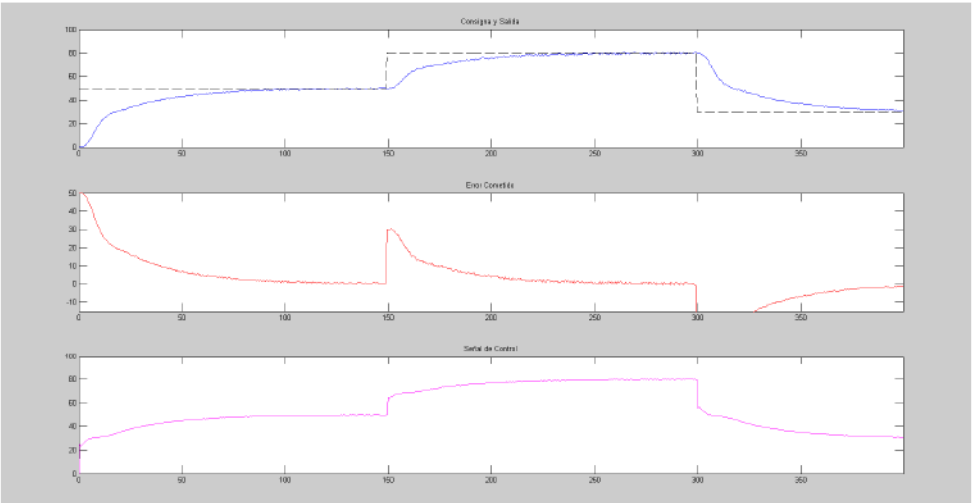


Figura 12.1.0.13 – Gráfica Ziegler-Nichols con Poca Sobreoscilación para un 60 % de punto de trabajo

- Ziegler-Nichols No Overshoot

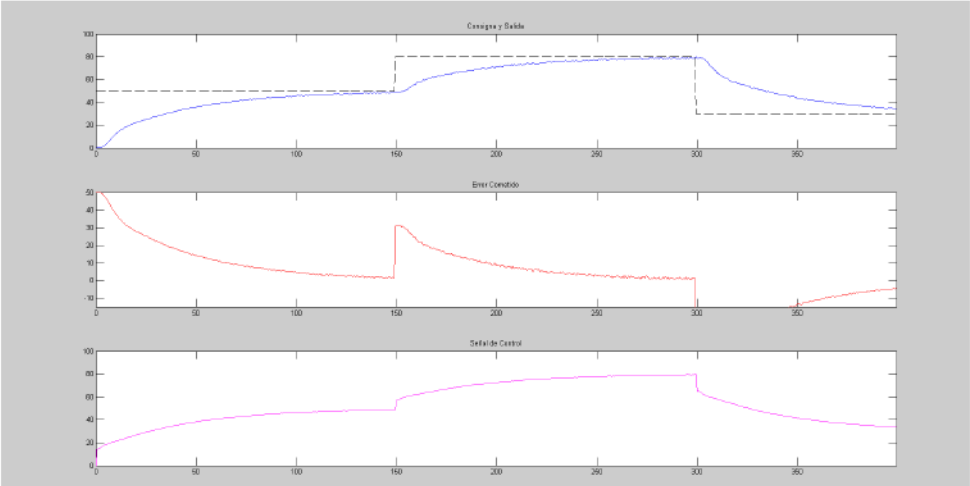


Figura 12.1.0.14 – Gráfica Ziegler-Nichols con No Overshoot para un 60 % de punto de trabajo

- Tyreus Luyben

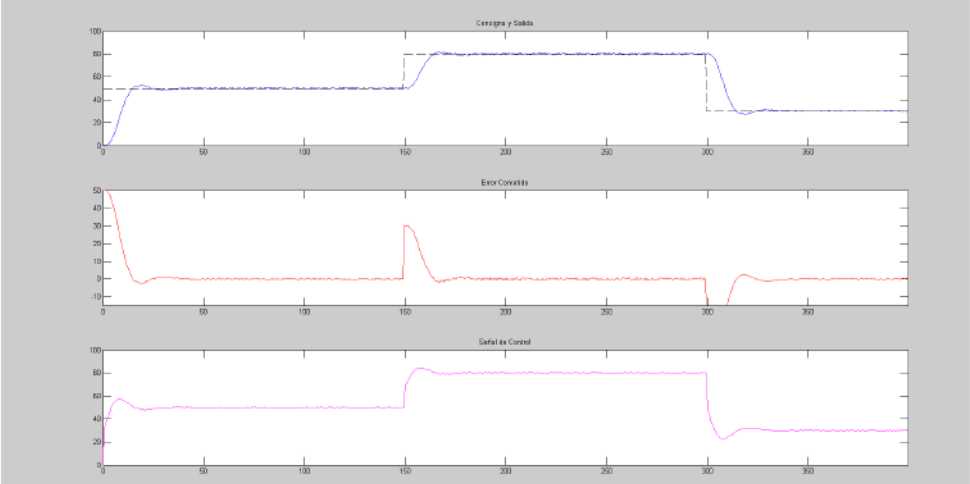


Figura 12.1.0.15 – Gráfica Tyreus-Luyben para un 60 % de punto de trabajo

4. Resultados para 80 % de punto de trabajo

- Relay-Feedback

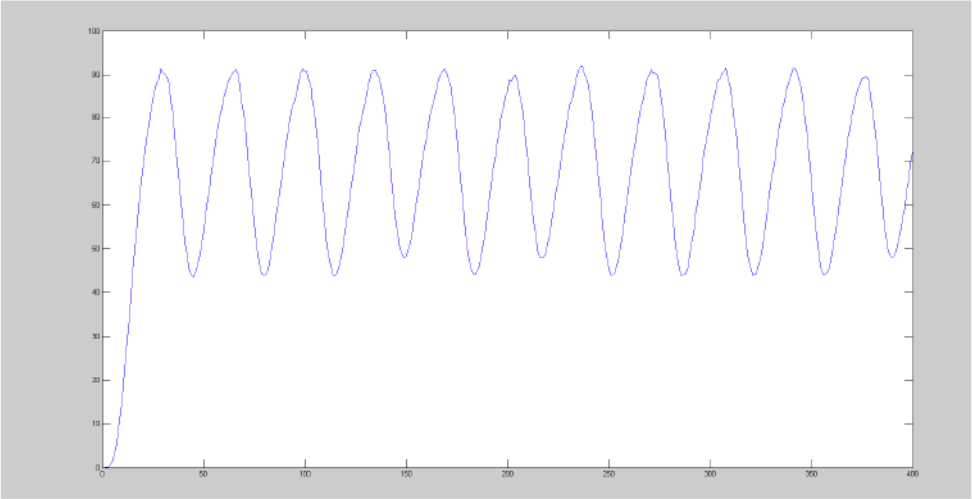


Figura 12.1.0.16 – Gráfica Relay-Feedback para punto de trabajo 80 %

- Ziegler-Nichols

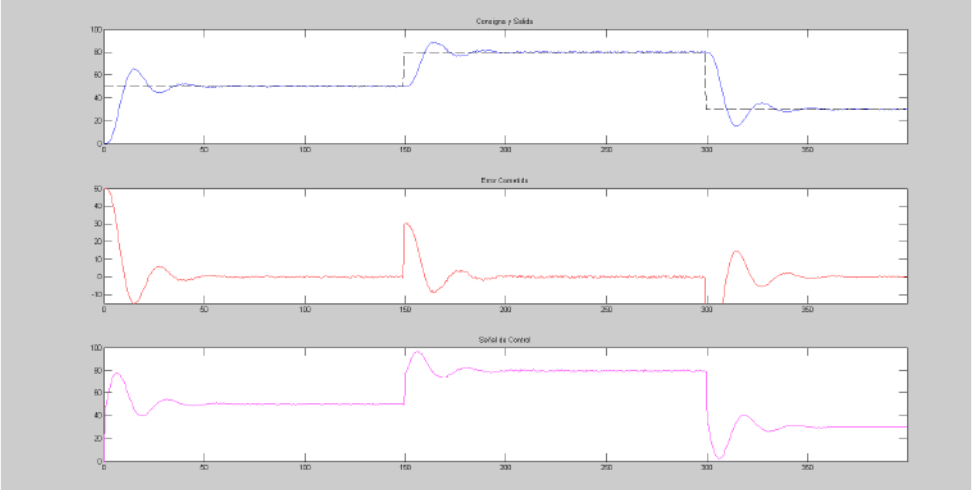


Figura 12.1.0.17 – Gráfica Ziegler-Nichols para un 80 % de punto de trabajo

- Ziegler-Nichols con Poca Sobreoscilación

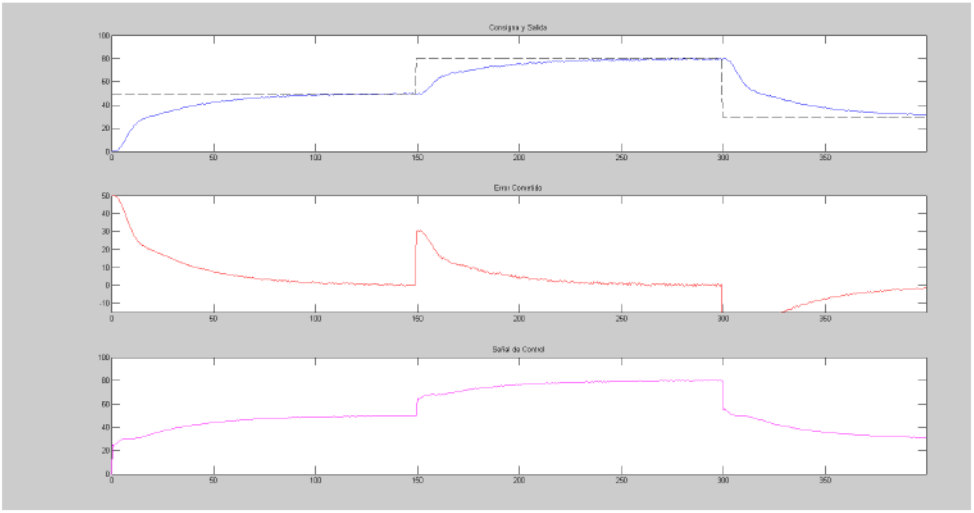


Figura 12.1.0.18 – Gráfica Ziegler-Nichols con Poca Sobreoscilación para un 80 % de punto de trabajo

- Ziegler-Nichols No Overshoot

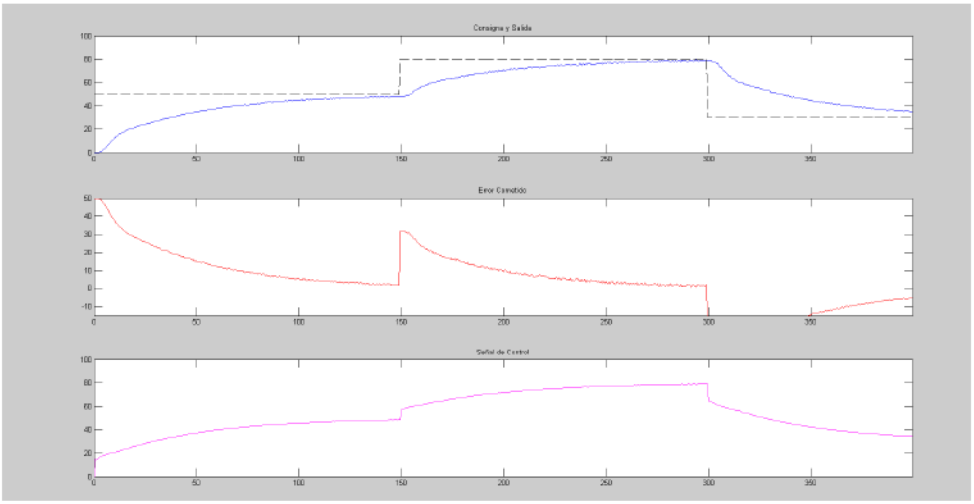


Figura 12.1.0.19 – Gráfica Ziegler-Nichols con No Overshoot para un 80 % de punto de trabajo

- Tyreus Luyben

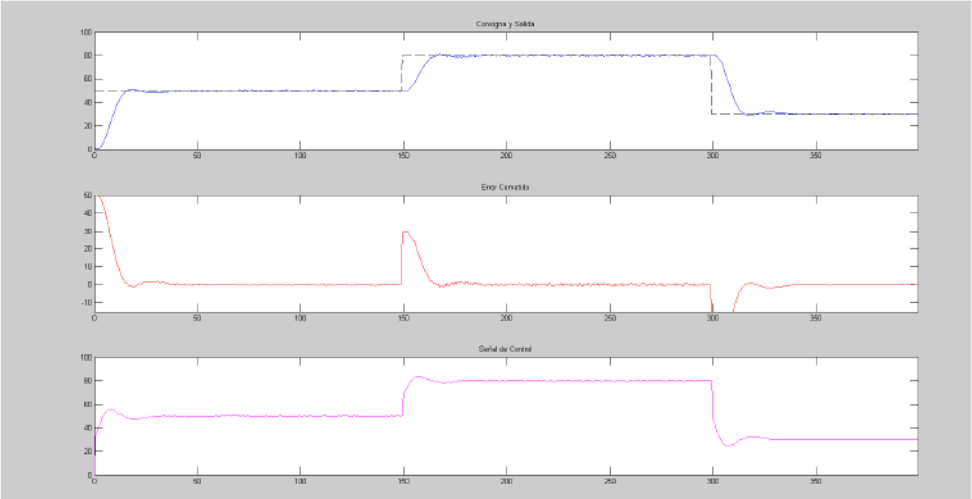


Figura 12.1.0.20 – Gráfica Tyreus-Luyben para un 80 % de punto de trabajo

Todos los resultados numéricos obtenidos para llevar a cabo estas simulaciones para los diferentes puntos de trabajo se puede consultar en las tablas 11.1.1.1, **??** y **??**.

12.2. Resultados del PID Adaptativo

Para el PID adaptativo no hay diferentes casos de modos de trabajo como en el PID estándar, ya que es una identificación en tiempo real. Por lo que los resultados a mostrar serán diferentes simulaciones con diferentes puntos de consigna y números de muestras.

- Para 1200 muestras

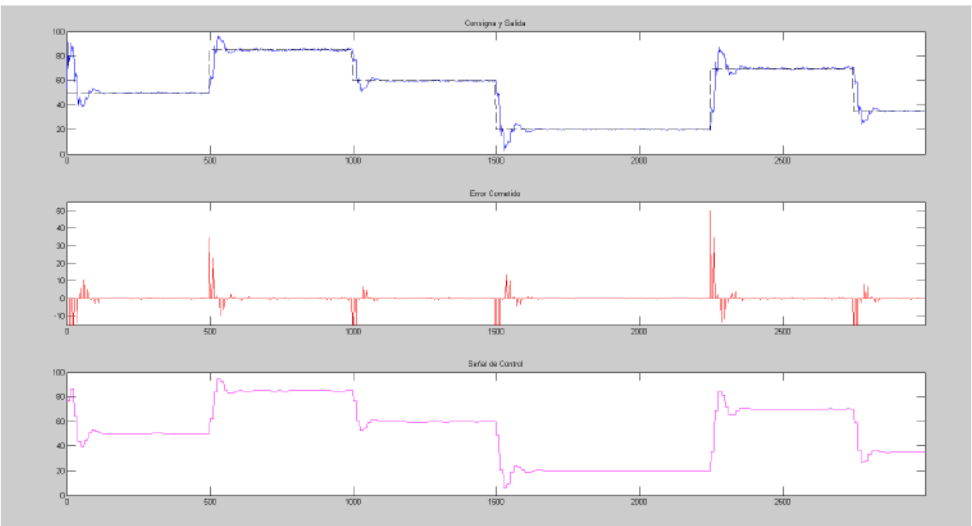


Figura 12.2.0.21 – PID Adaptativo con 1200 muestras

- Para 800 muestras

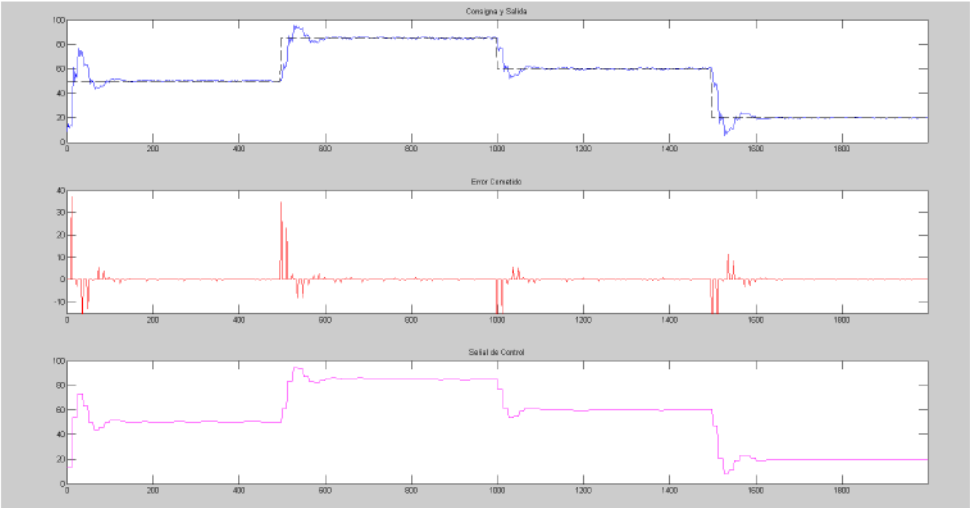


Figura 12.2.0.22 – PID Adaptativo con 800 muestras

13 Códigos de Programación

13.1. Scripts DAQ

Ya que las pruebas se han tenido que realizar por simulación debido a problemas antes comentados, a continuación se muestran los $Scripts$, facilitados por los tutores.

En todos estos $Scripts$ está implementado el códido correspondiente para trabajar con:

- NI_DAQ: Tarjeta National Instruments.
- Arduino_DAQ: Arduino como DAQ.
- Entorno de simulación

En todas estas funciones se indica qué parte del código serviría para cada uno de los casos anterior. En el presente proyecto se ha realizado mediante un entorno de simulación por problemas ya explicados anteriormente, por lo que el resto del código está comentado.

13.1.1. DAQ_Start

```
function [] = DAQ_Start()
% This code is used when you want to use a real NI_DAQ
%{
% This function configures the DAQ input and output

global AI
global AO

Device=daqhwinfo('nidaq'); % Find the desired device
ID=Device.InstalledBoardIds{1}; % Select the correct device from a list

AI=analoginput('nidaq',ID); % Inicialized of the input channel
AO=analogoutput('nidaq',ID); % Inicialized of the output channel

Input_channel=addchannel(AI,0); % Configure input channel (AI-0)
Input_channel.InputRange=[-10 10]; % Voltage input range
Input_channel.UnitsRange=[-100 100]; % Internal input range
Input_channel.Units='%'; % Internal units for input
```

```
Output_channel=addchannel(AO,0); % Configure output channel AO-0
Output_channel.OutputRange=[0 5]; % Voltage output range
Output_channel.UnitsRange=[0 100]; % Internal output range
Output_channel.Units='%'; % Internal units for output
%}

% This code is used when you want to use a real Arduino_DAQ
%{
% This function configures the DAQ input and output

global Arduino_DAQ

% The COM number depends on each system; it is neccesary to confirm in the
% "Devices administrator"
Arduino_DAQ=arduino('COM3');

% It's going to configure one output (pin 9); the analog input it
% is not neccesary to configure
Arduino_DAQ.pinMode(9,'output');
%}

% This code is used when you want to simulate a system

global Sim_Num % Numerator of the simulated system
global Sim_Den % Denominator of the simulated system
global Sim_Input % Input vector of the simulated system
global Sim_Output % Ouput vector of the simulated system

Ts=0.5; % Sample Time of the system

Type='NIT'; % Simulated system type

% Types available:
% FO ------> First order
% SO ------> Second order
% I -------> Integral
% FOI -----> First order with integration
% OSC -----> Oscillatory
% IR ------> Inverse response
% NIT -----> Non-interacting tanks
% FOTD ----> First order time delayed
% SOTD ----> Second order time delayed
% ITD -----> Integral time delayed
% FOITD ---> First order with integration time delayed
% OSCTD ---> Oscillatory time delayed
% IRTD ----> Inverse response time delayed
% NITTD ---> Non-interacting tanks time delayed
% HE ------> Heat exchanger

% The Numerator and Denominator of the system selected is stored
```

```
[Sim_Num,Sim_Den]=System_simulation(Type,Ts);

% The variables for the Inputs and Outputs of the simulated system are created
Sim_Input=zeros(1,length(Sim_Num));
Sim_Output=zeros(1,length(Sim_Den));

end
```

13.1.2. DAQ_Read

```
function [Input_data]=DAQ_Read()
% This code is used when you want to use a real NI_DAQ
%{
% This function read the value in the configured input in the DAQ

global AI

Input_data=getsample(AI);
%}

% This code is used when you want to use a real Arduino_DAQ
%{
% This function read the value in the configured input in the DAQ

global Arduino_DAQ

Input_data=Arduino_DAQ.analogRead(0);
% Arduino reading range is 0@1023, them the measure is scaled to 0@100
Input_data=100*Input_data/1023;
%}

% This code is used when you want to simulate a system

global Sim_Num  % Numerator of the simulated system
global Sim_Den  % Denominator of the simulated system
global Sim_Input  % Input vector of the simulated system
global Sim_Output  % Ouput vector of the simulated system

% The Output vector to simulate the selected system is update with the new value
Sim_Output(2:length(Sim_Output))=Sim_Output(1:end-1);
% Calculated with the Input vector update in the DAQ_Write function
Sim_Output(1)=(Sim_Input*Sim_Num'-Sim_Output(2:end)*Sim_Den(2:end)')/Sim_Den(1);

% The output value is ajusted to its minimum and maximum values
if (Sim_Output(1)>100)
    Sim_Output(1)=100;
elseif (Sim_Output(1)<0)
    Sim_Output(1)=0;
end
```

```
Noise=0.01;  % Definition of the maximum measurement noise

% As a normal sensor has noise that could be positive of negative, a random
% noise is simulated with the maximum value selected
Input_data=Sim_Output(1)+2*(0.5-rand)*Noise*Sim_Output(1);

end
```

13.1.3. DAQ_Write

```
function [] = DAQ_Write(Output_data_0)
% This code is used when you want to use a real NI_DAQ
%{
% This function write de desired value in the configured DAQ output

global AO

% Ensure that the value is inside the operation range
if Output_data_0>100
    Output_data_0=100;
elseif Output_data_0<0
    Output_data_0=0;
end

putsample(AO,Output_data_0);
%}

% This code is used when you want to use a real Arduino_DAQ
%{
% This function write de desired value in the configured DAQ output

global Arduino_DAQ

% Ensure that the value is inside the operation range
if Output_data_0>100
    Output_data_0=100;
elseif Output_data_0<0
    Output_data_0=0;
end

% As the range of the values is 0@100, the values are scaled to 0@255
Output_data_0=round(255*Output_data_0/100);

% Once the values are in the correct scale, they are "sent" to the DAQ
Arduino_DAQ.analogWrite(9,Output_data_0);
%}

% This code is used when you want to simulate a system

global Sim_Num % Numerator of the simulated system
global Sim_Input % Input vector of the simulated system

% The Input vector to simulate the selected system is update with the new value
Sim_Input(2:length(Sim_Input))=Sim_Input(1:end-1);
Sim_Input(1)=Output_data_0;

end
```

13.1.4. DAQ_Stop

```
function [] = DAQ_Stop()
% This code is used when you want to use a real NI_DAQ
%{
% This function is used to finaliced the uses of the DAQ

global AI
global AO

% Clear the input channel
stop(AI);
delete(AI);

% Clear the output channel
putsample(AO,0); % Ensure the output channel is resetted
stop(AO);
delete(AO);
%}

% This code is used when you want to use a real Arduino_DAQ
%{
% This function is used to finaliced the uses of the DAQ

global Arduino_DAQ

% Clear the output channel
Arduino_DAQ.analogWrite(9,0); % Ensure the output channel is resetted

% The variable of the DAQ is deleted
delete(Arduino_DAQ);
clear Arduino_DAQ;
% And the port is liberated
delete(instrfind({'Port'},{'COM3'}))
%}

% This code is used when you want to simulate a system

global Sim_Num % Numerator of the simulated system
global Sim_Den % Denominator of the simulated system
global Sim_Input % Input vector of the simulated system
global Sim_Output % Ouput vector of the simulated system

clear Sim_Num
clear Sim_Den
clear Sim_Input
clear Sim_Output

end
```

13.2. Sistema de Simulación

Al haber realizado las pruebas en entorno de simulación, además de los $Scripts$ anteriores, se deben crear un entorno de simulación con los diferentes tipos de sistemas,

En este caso, se tendrá un sistema de segundo orden para emular la planta real.

El sistema simulado será 'NIT' y el PID estándar y PID adaptativo han sido simulados con dos funciones de transferencia diferentes que se indican en el código.

```
function [Num,Den] = System_simulation(Type,Ts)
% This function gives the discrete transfer function of a system
%         Num(z)      a(0)*z^m + a(1)*z^(m-1) +...+ a(m-1)*z + a(m)
% G(z) = --------- = ------------------------------------------------
%         Den(z)      b(0)*z^n + b(1)*z^(n-1) +...+ b(n-1)*z + b(n)
%

Delay=0.5;  % Definition of the simulated delay for "time delayed" systems

if strcmp(Type,'FO')
    % First order
    G=tf(1,[0.1 1]);
    G=c2d(G,Ts,'tustin');
    Num=G.num{1};
    Den=G.den{1};

elseif strcmp(Type,'SO')
    % Second order
    G=tf(1,[1 1 1]);
    G=c2d(G,Ts,'tustin');
    Num=G.num{1};
    Den=G.den{1};

elseif strcmp(Type,'I')
    % Integral
    G=tf(0.6,[1 0]);
    G=c2d(G,Ts,'tustin');
    Num=G.num{1};
    Den=G.den{1};

elseif strcmp(Type,'FOI')
    % First order with integration
    G=tf(0.6,[0.1 1 0]);
    G=c2d(G,Ts,'tustin');
    Num=G.num{1};
    Den=G.den{1};

elseif strcmp(Type,'OSC')
    % Oscillatory
    G=tf(1,[0.005 0.06 1]);
```

```
    G=c2d(G,Ts,'tustin');
    Num=G.num{1};
    Den=G.den{1};

elseif strcmp(Type,'IR')
    % Inverse response
    G=tf([-0.0625 1],[0.015625 0.125 1]);
    G=c2d(G,Ts,'tustin');
    Num=G.num{1};
    Den=G.den{1};

elseif strcmp(Type,'NIT')
    % Non-interacting tanks
     %G=tf(1,[0.025 1]);  % FT PID Adaptativo
    G=tf(1,[2 1]);       % FT PID Estandar
    G=G^4;
    G=c2d(G,Ts,'tustin');
    Num=G.num{1};
    Den=G.den{1};

elseif strcmp(Type,'FOTD')
    % First order time delayed
    G=tf(1,[0.1 1]);
    G=c2d(G,Ts,'tustin');
    Num=G.num{1};
    Num=[zeros(1,round(Delay/Ts)) Num];
    Den=G.den{1};

elseif strcmp(Type,'SOTD')
    % Second order time delayed
    G=tf(1,[0.025 0.03 1]);
    G=c2d(G,Ts,'tustin');
    Num=G.num{1};
    Num=[zeros(1,round(Delay/Ts)) Num];
    Den=G.den{1};

elseif strcmp(Type,'ITD')
    % Integral time delayed
    G=tf(0.6,[1 0]);
    G=c2d(G,Ts,'tustin');
    Num=G.num{1};
    Num=[zeros(1,round(Delay/Ts)) Num];
    Den=G.den{1};

elseif strcmp(Type,'FOITD')
    % First order with integration time delayed
    G=tf(0.6,[0.1 1 0]);
    G=c2d(G,Ts,'tustin');
    Num=G.num{1};
    Num=[zeros(1,round(Delay/Ts)) Num];
```

```
    Den=G.den{1};

elseif strcmp(Type,'OSCTD')
    % Oscillatory time delayed
    G=tf(1,[0.005 0.06 1]);
    G=c2d(G,Ts,'tustin');
    Num=G.num{1};
    Num=[zeros(1,round(Delay/Ts)) Num];
    Den=G.den{1};

elseif strcmp(Type,'IRTD')
    % Inverse response time delayed
    G=tf([-0.0625 1],[0.015625 0.125 1]);
    G=c2d(G,Ts,'tustin');
    Num=G.num{1};
    Num=[zeros(1,round(Delay/Ts)) Num];
    Den=G.den{1};

elseif strcmp(Type,'NITTD')
    % Non-interacting tanks time delayed
    G=tf(1,[0.025 1]);
    G=G^4;
    G=c2d(G,Ts,'tustin');
    Num=G.num{1};
    Num=[zeros(1,round(Delay/Ts)) Num];
    Den=G.den{1};

elseif strcmp(Type,'HE')
    % Heat exchanger
    G=tf(1,[0.1 0]);
    G=c2d(G,Ts,'tustin');
    Num=G.num{1};
    Num=[Num zeros(1,round(Delay/Ts))]-[zeros(1,round(Delay/Ts)) Num];
    Den=G.den{1};

else
    % Non type
    G=tf(1);
    G=c2d(G,Ts,'tustin');
    Num=G.num{1};
    Den=G.den{1};

end

end
```

13.3. Regulador PID Estándar

13.3.1. Programa principal para la obtencion del PID estándar

```
clc
clear

%% INICIALIZACIÓN DE VARIABLES GLOBALES

T=0.5;  % Periodo de muestreo segundos
t=200*2;  % Tiempo de la prueba (segundos)
n=floor(t/T);

% Inicialización de las variables necesarias como vectores
Vector_SP=zeros(n,1);
Entrada=zeros(n,1);
Salida=zeros(n,1);
Plot_E=zeros(n,1);
Tiempo=zeros(n,1);
E=zeros(1,3);

% Eleccion del punto de trabajo

fprintf('Elija el punto de trabajo: \n 1) 40 \n 2) 50 \n 3) 60 \n 4) 80\n ')
p_trabajo=input('');

% En funcion del punto de trabajo se obtienen valores diferentes de Tc y
% Kc (calculados mediante la funcion Relay-Feedback)

[Tc,Kc]=punto_trabajo(p_trabajo);

% Eleccion del modo de trabajo

fprintf('Elija el modo de trabajo:\n 1)Z&N  \n 2)Poca Sobre  \n 3)NO Over \n 4)Tyreus Luyben\n')
modo_trabajo=input('');

% En funcion del modo de trabajo se obtienen valores diferentes de Kp, Td,
% Ti

[Kp,Kd,Ki]=parametros(Kc,Tc,modo_trabajo,T);

% Valor consigna
Vector_consigna=[50,50,80,80,30,30,65,65,40,40,75,75,20,20];
SP=Vector_consigna(1,1);

% Inicializacion señal de control
U=0;
j=1;
```

```
% % Inicializamos la variable de entrada y final de nuestro eje x

prin=0;
f=((n*T)-T);
y=0;

DAQ_Start; % Activación de la TAD


% Se controla la temporización

tic % Activación del control del tiempo
t1=toc;

%% BUCLE PRINCIPAL

for i=1:n

    DAQ_Write(U); % Se escribe la señal de control
    PV=DAQ_Read; % Se lee la variable de proceso de la planta

    % Se almacenan los datos

    Entrada(i)=U;
    Salida(i)=PV;


    %Comprobamos si ha cambiado el valor de la consigna


    if rem(i,100)==0

       SP=Vector_consigna(1,j+1);
       j=j+1;
    end

    % Vector que guarda los valores de consigna
    Vector_SP(i)=SP;

    % Antes de realizar el cálculo de la señal de control hay que
    % recalcular los valores del error

    E(2:end)=E(1:end-1)
    E(1)=SP-PV;
    Plot_E(i)=E(1);

    % Cálculo de la señal de control del PID

    [U]=Regulador_PID(Kp,Ki,Kd,T,U,E);
```

```
    % GRAFICAR

    a=subplot(3,1,1);
    plot(y,Salida(1:i))
    title(a,'Consigna y Salida')
    axis([prin,f,0,100])
    hold on
    plot(y,Vector_SP(1:i), 'k--')
    hold off;
    b=subplot(3,1,2);
    plot(y,Plot_E(1:i), 'r')
    title(b,'Error Cometido')
    axis([prin,f,-15,SP+20])
    c= subplot(3,1,3);
    plot(y,Entrada(1:i), 'm')
    title(c,'Señal de Control')
    axis([prin,f,0,100])
    y=[y,y(end)+T];


    % TEMPORIZACION

        t2=toc;

        if (t2-t1)<T

            pause(T-t2+t1)

        else % Si se incumple la temporización, se aborta la prueba
             warning('Se ha incumplido el tiempo de muestreo');
        end

        Tiempo(i)=toc;
        t1=t2;



end


%% Finalización de procesos

% Finalizamos la comunicación con la tarjeta de adquisición de datos
DAQ_Stop;
```

13.3.2. Método Relay-Feedback

```
function Output = Relay_FeedBack(SP,h,Ts)
% Funcion para obtener la respuesta de un sistema al realizar una prueba de
% Relay FeedBack.

% La funcion devuelve la salida del sistema para un punto de trabajo (SP), y
% un ancho de ventana de histeresis (d) dado; hay que indicar tambien el periodo
% de muestreo (Ts).

% Inicializacion de la tarjeta de adquisicion de datos
DAQ_Start;

% Inicializacion de la variable de salida y del tiempo de la prueba
Time=200; % Tiempo de la prueba en segundos
n=floor(Time/Ts); % Numero de veces que se repetira el bucle
Output=zeros(n,1);

% Inicializacion de las temporizaciones
tic
t1=toc;

% Bucle principal
for i=1:1:n
  % Comunicacion con la tarjeta de adquisicion de datos
  Output(i)=DAQ_Read; % Lectura de la salida actual de la planta
  % La señal de control a la planta dependera¡ de la entrada leida
  if Output(i)<=SP-h
    Input=100;
  elseif Output(i)>SP+h
    Input=0;
  end
  DAQ_Write(Input); % Escritura de la señal de control correspondiente

  % Control de la temporizacion del bucle
  t2=toc;
  if (t2-t1)>Ts
    warning('Se ha superado el tiempo de muestro en la iteracionn: %d',i)
  else
    pause(Ts-t2+t1);
  endif
  t1=toc;

 end

% Desconexion de la TAD
DAQ_Stop

end
```

13.3.3. Calculo de los parámetros del regulador

```
% Funcion para el calculo de los parametros del regulador
% Devuelve los valores de los parametros Kp,Ti y Td
% A la funcion se le pasan los parametros Kc y Tc (calculados con el
% Relay-Feedback) y el modo de trabajo del PID ( 1)Z&N  \n 2)Poca Sobre  \n 3)NO Over
% \n 4)Tyreus Luyben)

function [Kp,Ti,Td]= parametros(Kc,Tc,Tipo)

    % Inicializacion de los parametros
    Kp=0;
    Td=0;
    Ti=0;

    switch Tipo

            case 1

                Kp=0.6*Kc;
                Ti=0.5*Tc;
                Td=0.125*Tc;

             case 2

                Kp=0.33*Kc;
                Ti=Tc/2;
                Td=Tc/3;

             case 3

                Kp=0.2*Kc;
                Ti=Tc;
                Td=Tc/3;

             case 4

                Kp=0.45*Kc;
                Ti=2.2*Tc;
                Td=Tc/6.3;

    end

end
```

13.3.4. Cálculo de la señal de control

```
%% FUNCIÓN PARA CALCULAR LA SEÑAL DE CONTROL DE UN PID DISCRETO

function [U]=Regulador_PID(Kp,Ki,Kd,T,U,E)

% Se parte de la última señal de control calculada y el valor del error
% acutal y anteriores
% E(z)=[E(z); E(z-1); E(z-2)]

% ´Se calcula la señal de control mediante el método de Tustin que parte de
% la función de transferencia discretizada
%  U(z)              T   z+1        z-1
% ------ = Kp + Ki·---·------ + Kd·------
%  E(z)              2   z-1        z·T

% Y operando se llega a una ecuación de la señal de control reducida
% u{k}=u{k-1} + q0*e{k} + q1*e{k-1} + q2*e{k-2}

% Sabiendo que:

%             Td
% q0 = Kp(1+ ----)
%             T
%              Td      T
% q1 = Kp(-1 -2·-- + ----)
%                T      Ti
%         Td
% q2 = Kp(----)
%         T

% Y:
%        Kp
% Ti =  ----
%        Ki

%        Kd
% Td =  ----
%        Kp

q0=Kp+((Ki*T)/2)+(Kd/T);
q1=((Ki*T)/2)-Kp-((2*Kd)/T);
q2=Kd/T;

% Señal de control indicada arriba con el valor de los parámetros 'q0',
% 'q1' y 'q2'

 U=U+q0*E(1)+q1*E(2)+q2*E(3);
```

```
% Hay que limitar los valores máximos y mínimos del sistema

if U>100
    U=100;
elseif U<0
    U=0;
end

end
```

13.4. Código PID Adaptativo

13.4.1. Programa principal

```
clc
clear

%% Inicialización de variables

% Globales

T=0.5;  % Periodo de muestreo segundos
t=100*2;  % Tiempo de la prueba (segundos)
n=floor(t/T);
n_RLS=5;
T_n_RLS=T*n_RLS;

% Se asigna un vector con varios valores de consigna

Vector_consigna=[50,50,80,80,65,20,20,20,80,80,35,35,35,70,70,70,40,40,60,60];
j=1;

SP=Vector_consigna(1,1);


% Variables para el RLS

% Se parte de una función de transferencia genérica

%           z^m + b1·z^(m-1) + b2·z^(m-2) + ... + bm-1·z + bm
%   G(z) = ------------------------------------------------------
%           z^n + a1·z^(n-1) + a2·z^(n-2) + ... + an-1·z + an

% Nos quedamos con grado 1 para el numerador y grado 2 para el denominador
% Calculamos los parámetros iniciales de G(z)
%
%

%%

% Se comprueba si el valor de consigna ya ha sido un valor anterior
% si es así los datos se recogen del fichero en vez de dar valores
% aleatorios a los parámetros

 if SP==40 || SP==50 || SP==60 || SP==80

    [Num,Den,E]=Base_Datos(SP);
    E=rand(3,1)*100;
```

```
 else

%Si no es así generamos un valor aleatorio del error, ya que la primera vez
%que calculamos
%la señal de control necesitamos comparar el error actual con el anterior

E=rand(3,1)*100;

Num=rand(1,2);  % Inicializamos el numerador con números aleatorios

Den=rand(1,3);  % Inicializamos el denominador con números aleatorios

% La función a calcular es del siguiente modo

%                 b1·z
%   G(z) = ———————————————————
%          z^2 + a1·z + a2

Num(end)=0;
Den(1)=1;
 end

% Hay que desarrollar la función de transferencia para obtener la ecuación
% en diferencias

%   y{k} = b1·u{k-1} - a1·y{k-1} - a2·y{k-2}

%% Inicializacion de los parametros para el RLS

theta=[Num(1:end-1)'; Den(2:end)'];  % Se define theta de forma genérica
% Se define el vector de regresión x (debe tener el mismo tamaño que theta)
X=rand(length(theta),1);
% Se define la matriz P, utilizada por RLS para el cálculo de los parámetros
P=diag(rand(1,length(theta)));

%% Inicializacion de las variables necesarias para el proceso

% Hay que asignar valores de entrada y salida con números aleatorios, ya
% que la primera vez este valor no se puede calcular

Entrada_ant=rand(1,1)*100;
Salida_ant=rand(1,1)*100;


% Asignamos a la señal de ccontrol 'U' el valor de Entrada_ant, ya que es
% lo que vamos a escribir

U=Entrada_ant;
```

```
% Se inicializan una serie de vectores para guardar los datos obtenidos en cada
% muestra
Tiempo=zeros(n,1);
Entrada=zeros(n,1);
Salida=zeros(n,1);
Plot_E=zeros(n,1);


Numerador=zeros(floor(n/n_RLS),2);
Denominador=zeros(floor(n/n_RLS),3);
Vector_SP=zeros(n,1);

%Inicializamos la variable de entrada y final de nuestro eje x

prin=0;
f=((n*T_n_RLS)-T_n_RLS);
y=0;

%% Apertura del fichero


 fileID=fopen('Num_Dem.txt','wt');
 file=fopen('Error.txt','wt');

% Se inicializa la TAD

DAQ_Start; % Activación de la TAD


% Se controla la temporización

tic % Activación del control del tiempo
t1=toc;

%% BUCLE PRINCIPAL

for i=1:n

    DAQ_Write(U); % Se escribe la señal de control
    PV=DAQ_Read; % Se lee la variable de proceso de la planta

    % Se almacenan los datos

    Entrada(i)=U;
    Salida(i)=PV;


    %% Se calcula el RLS y los parametros del regulador y señal de control
    % cada cierto numero de muestras, de este modo se filtra la señal
```

```
if rem(i,n_RLS)==0

    %% Calculo del RLS y llamada a su funcion para obtener los
    % parametros de la funcion de transferencia

     Salida_act=sum(Salida(i+1-n_RLS:i))/n_RLS;
     Entrada_ant=sum(Entrada(i+1-n_RLS:i))/n_RLS;

     % Sabiendo que X = [-Y(z-1); -Y(z-2); U(z-1)]

     X(1)=Entrada_ant;
     X(3)=X(2);
     X(2)=-Salida_ant;

     % Función RLS para calcular los nuevoa parámetros de theta

     [theta,P]=RLS(Salida_act,theta,X,P);

     % Se redefinen los valores para la siguiente muestra

     Salida_ant=Salida_act;
     Num=[theta(1:1)' 0];
     Den=[1 theta(2:end)'];
     Numerador(floor(i/n_RLS),:)=Num;
     Denominador(floor(i/n_RLS),:)=Den;

     %% Se escriben en el fichero los valores obtenidos de denominador y numerador

     fprintf(fileID,'%.4f\t%.4f\t%.4f\t%.4f\t%.4f', Numerador(floor(i/n_RLS),:),
     Denominador(floor(i/n_RLS),:));
     fprintf(fileID,'\n');


    %% Calculo de los parametros del regulador en funcion de la funcion de transferencia

     [Kp,Ki,Kd]=Parametros_PID(Num,Den,T_n_RLS);


     %Comprobamos si ha cambiado el valor de la consigna, si ha cambiado
     %y existe un fichero con datos de una prueba anterior para ese SP
     % se cogen los datos del fichero

     if rem(i,100)==0

        SP=Vector_consigna(1,j+1)
        j=j+1;

        Vector_SP(i)=SP;

        if (SP==40 && Vector_SP(i-1)~=40) || (SP==50 &&  Vector_SP(i-1)~=50) || (SP==60 &&
```

```
Vector_SP(i-1)~=60) || (SP==80 &&  Vector_SP(i-1)~=80)

            [Num,Den,E]=Base_Datos(SP);
          end

        end


        %% Calculo del error

        % Antes de realizar el cálculo de la señal de control hay que
        % recalcular los valores del error

        E(2:end)=E(1:end-1)
        E(1)=SP-PV;
        Plot_E(i)=E(1);

        %% Se escribe en el fichero el valor del error obtenido

         fprintf(file,'%.4f \n',Plot_E(i));

        %% Calculo de la señal de control

        [U]=Regulador_PID(Kp,Ki,Kd,T_n_RLS,U,E);

    end

    %% Se asigna al un nuevo vector de consignas el valor del SP

     Vector_SP(i)=SP;


    % GRAFICAR

    a=subplot(3,1,1);
    plot(y,Salida(1:i))
    title(a,'Consigna y Salida')
    axis([prin,f,0,100])
    hold on
    plot(y,Vector_SP(1:i), 'k--')
    hold off;
    b=subplot(3,1,2);
    plot(y,Plot_E(1:i), 'r')
    title(b,'Error Cometido')
    axis([prin,f,-15,SP+20])
    c= subplot(3,1,3);
    plot(y,Entrada(1:i), 'm')
    title(c,'Señal de Control')
    axis([prin,f,0,100])
    y=[y,y(end)+T_n_RLS];
```

```
    % TEMPORIZACION

        t2=toc;

        if (t2-t1)<T_n_RLS

            pause(T_n_RLS-t2+t1)

        else % Si se incumple la temporización, se aborta la prueba
             warning('Se ha incumplido el tiempo de muestreo');
        end

        Tiempo(i)=toc;
        t1=t2;



end

%% Al acabar de trabajar con los fichero hay que cerrarlos: cierre de ficheros

 fclose(fileID);
 fclose(file);


%% Finalización de procesos

% Finalizamos la comunicación con la tarjeta de adquisición de datos
DAQ_Stop;
```

13.4.2. Función RLS

```
%% FUNCIÓN RLS ( Cálculo de los parámetros de la función de transferencia de un sistema discreto)

function [theta,P]=RLS(Salida_n,theta,X,P)

% El algoritmo RLS es usado para calcular los coeficientes de theta, que
% serán los parámetros de la función de transferencia a partir de la
% ecuación

% y=X'*theta

% y== variable medida
% X== vector que se forma en función de las entradas y salidas
% proporcionadas por el sistema

%% Definición de parámetros del RLS
```

```
lambda=0.9;  % Factor de olvido

% Matriz K de ganancias

 K=(P*X)/(lambda+X'*P*X);

% Calculo del error

Error=Salida_n-X'*theta;

% Cálculo de los nuevos coeficientes de theta (parámetros de la función de
% transferencia)

theta=theta+K*Error;

% Despreciamos los valor menores a 1e-5 (valores absolutos)

 theta(abs(theta)<1e-5)=0;

% Cálculo de la matriz P para la siguiente iteración

P=1/lambda*(P-(K*X'*P));


end
```

13.4.3. Función Cálculo de los parámetros del regulador

```
%% FUNCIÓN PARA EL CÁLCULO DE LOS PARÁMETROS DEL PID (Ki,Kd,Kp) EN FUNCIÓN DE LA FUNCIÓN DE TRANSF

function [Kp,Ki,Kd]=Parametros_PID(Num,Den,T)

% Elementos del vector theta (parámetros de la función de transferencia)
b1=Num(1);
a1=Den(2);
a2=Den(3);

a=10;

p0=(1/b1)*(1+a1-(3/a));
p1=(1/b1)*(a2-a1+(3/a^2));
p2=-(1/b1)*(a2+(1/a^3));


% Calculo parametros regulador
Kp=-(p1+2*p2);

Kd=p2*T;

Ki=(p0+p1+p2)/T;

end
```

13.4.4. Función Cálculo de la señal de control del regulador

```
%% FUNCIÓN PARA CALCULAR LA SEÑAL DE CONTROL DE UN PID DISCRETO

function [U]=Regulador_PID(Kp,Ki,Kd,T,U,E)

% Se parte de la última señal de control calculada y el valor del error
% acutal y anteriores
% E(z)=[E(z); E(z-1); E(z-2)]

% ´Se calcula la señal de control mediante el método de Tustin que parte de
% la función de transferencia discretizada
%  U(z)              T   z+1        z-1
% ------ = Kp + Ki·---·------ + Kd·------
%  E(z)              2   z-1        z·T

% Y operando se llega a una ecuación de la señal de control reducida
% u{k}=u{k-1} + q0*e{k} + q1*e{k-1} + q2*e{k-2}

% Sabiendo que:
```

```
%              Td
% q0 = Kp(1+ ———)
%              T
%               Td       T
% q1 = Kp(—1 —2·—— + ———)
%                 T       Ti
%          Td
% q2 = Kp(———)
%          T

% Y:
%         Kp
% Ti =  ———
%         Ki

%         Kd
% Td =  ———
%         Kp
```

```
q0=Kp+((Ki*T)/2)+(Kd/T);
q1=((Ki*T)/2)—Kp—((2*Kd)/T);
q2=Kd/T;

% Señal de control indicada arriba con el valor de los parámetros 'q0',
% 'q1' y 'q2'

 U=U+q0*E(1)+q1*E(2)+q2*E(3);

% Hay que limitar los valores máximos y mínimos del sistema

if U>100
    U=100;
elseif U<0
    U=0;
end

end
```

13.4.5. Función para recoger los datos de un fichero a partir de un SP conocido

```
%% FUNCIÓN PARA OBTENER LOS VALORES DE LOS PARÁMETROS DESDE UN FICHERO PARA UN SP DETERMINADA

function [Num,Den]=Base_Datos(SP)

%% 40 SP
if SP==40

    % Numerador y Denominador
```

```
    fileID=fopen('Num_Dem40.txt') % Se carga el archivo para ese SP

  % Se crea un bucle para leer la ultima fila del fichero

    while feof(fileID)~=1
    tline=fgetl(fileID)
    end

    fclose(fileID) % Se cierra el fichero

    % tline nos devulve 'char', por lo que creamos un vector para guardar
    % los datos en formato 'float'

    Vector_Num_Den=str2num(tline)

    Num=Vector_Num_Den(1,1:2);

    Den=Vector_Num_Den(1,3:5);


%% 50 SP

elseif SP==50

   % Numerador y Denominador

    fileID=fopen('Num_Dem50.txt') % Se carga el archivo para ese SP

    % Se asignan a los parámetros los últimos valores del fichero

    while feof(fileID)~=1
    tline=fgetl(fileID)
    end

    fclose(fileID)

    Vector_Num_Den=str2num(tline)

    Num=Vector_Num_Den(1,1:2);

    Den=Vector_Num_Den(1,3:5);


%% 60 SP
elseif SP==60

  % Numerador y Denominador

    fileID=fopen('Num_Dem60.txt') % Se carga el archivo para ese SP
```

```
    % Se asignan a los parámetros los últimos valores del fichero

    while feof(fileID)~=1
    tline=fgetl(fileID)
    end

    fclose(fileID)

    Vector_Num_Den=str2num(tline)

    Num=Vector_Num_Den(1,1:2);

    Den=Vector_Num_Den(1,3:5);



%% 80 SP
else

   % Numerador y Denominador

    fileID=fopen('Num_Dem80.txt')  % Se carga el archivo para ese SP

    % Se asignan a los parámetros los últimos valores del fichero

    while feof(fileID)~=1
    tline=fgetl(fileID)
    end

    fclose(fileID)

    Vector_Num_Den=str2num(tline)

    Num=Vector_Num_Den(1,1:2);

    Den=Vector_Num_Den(1,3:5);

end

end
```

TÍTULO: **IMPLEMENTACIÓN DE UN ALGORITMO DE CONTROL ADAPTATIVO**

PLIEGO DE CONDICIONES

PETICIONARIO: **ESCUELA UNIVERSITARIA POLITÉCNICA**

AVDA. 19 DE FEBREIRO, S/N

15405 - FERROL

FECHA: JUNIO DE 2019

AUTOR: EL ALUMNO

Fdo.: MARÍA REFOJO FERREIRÓS

Índice del documento PLIEGO DE CONDICIONES

14 Pliego de Condiciones

14.1. Condiciones de Trabajo

Las pruebas realizadas en la planta de control del "Laboratorio de Optimización y Control.en el presente proyecto, se han realizado en un entorno limpio, seco y a temperatura ambiente.

Si se debiese realizar algún tipo de conexionado con la placa Arduino UNO, debería llevarse a cabo con la planta apagada.

La planta se encuentra sobre una superficie nivelada para evitar la pérdida de líquido. Éste líquido ha de ser de baja densidad para no perturbar el funcionamiento de las bombas y es recomendable el uso de líquidos similares al agua, en este caso agua, para evitar la corrosión.

14.2. Componentes de Sustitución

Si fuese necesaria la sustitución de un componente de la planta debido a un deterioro o avería, éste debe ser sustituído por uno con las mismas características que el anterior.

La sustitución del componente averiado deberá hacerse con la planta desconectada, al igual que se comentó con las conexiones con la tarjeta de adquisición de datos.

14.3. Software

Para el uso de la planta se ha utilizado la versión de $MatLab$ R2014a, pudiendo utilizar otras versiones posteriores.

14.4. Comprobaciones para la puesta en marcha

La red eléctrica a la que está conectada la planta de control debe tener una tensión de alimentación de 230 V.

Una vez realizada la puesta en marcha de la planta, debe comprobarse que la comunicación entre la planta y el PC se ha realizado de forma correcta. Para realizar esta comunicación, se debe detectar el puerto de conexión del Arduino UNO con el PC y cargar unos $Scripts$ que permiten la comunicación.

TÍTULO: **IMPLEMENTACIÓN DE UN ALGORITMO DE CONTROL ADAPTATIVO**

ESTADO DE MEDICIONES

PETICIONARIO: **ESCUELA UNIVERSITARIA POLITÉCNICA**

AVDA. 19 DE FEBREIRO, S/N

15405 - FERROL

FECHA: JUNIO DE 2019

AUTOR: EL ALUMNO

Fdo.: MARÍA REFOJO FERREIRÓS

Índice del documento ESTADO DE MEDICIONES

15 Estado de Mediciones

15.1. Mano de Obra

Tarea	Tiempo (h)
Programación PID Estándar	40
Estudio y Programación del RLS	80
Programación PID Adaptativo	120
Memoria	220
Total	480

15.2. Amortizaciones

CONCEPTO	CANTIDAD
Ordenador	1
Licencia MatLab	1

TÍTULO: **IMPLEMENTACIÓN DE UN ALGORITMO DE CONTROL ADAPTATIVO**

PRESUPUESTO

PETICIONARIO: **ESCUELA UNIVERSITARIA POLITÉCNICA**

AVDA. 19 DE FEBREIRO, S/N

15405 - FERROL

FECHA: JUNIO DE 2019

AUTOR: EL ALUMNO

Fdo.: MARÍA REFOJO FERREIRÓS

Índice del documento PRESUPUESTO

16 Presupuesto

16.1. Mano de Obra

CONCEPTO	UNIDADES	PRECIO UNITARIO	TOTAL
Mano de obra Graduado en Ingeniería electrónica Industrial y Automática	480 horas	50 €	24000 €
Total			24000 €

16.2. Amortizaciones

CONCEPTO	PRECIO	AMORTIZACIÓN TOTAL	AMORTIZACIÓN EN 3 MESES	PRECIO
Ordenador	1000 €	5 años	4.931 %	49.31 €
Licencia MatLab	2000 €	33 % por año	8.137 %	162.74 €
Total				212.05 €

16.3. Presupuesto Total

CONCEPTO	PRECIO
Materiales, mano de obra y elementos auxiliares	24000 €
Amortizaciones	212.05 €
IVA 21 %	5084.53 €
Total	29205.58 €